RADC-TR-80-365
Final Technical Report
December 1980

CO-CHANNEL INTERFERENCE SEPARATION

Pattern Analysis and Recognition Corporation

Dr. Robert J. Dick

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, New York 13441

This report has been reviewed by the RADC Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-80-365 has been reviewed and is approved for publication.

APPROVED: Melvin G. Manor Jr.

MELVIN G. MANOR, JR.
Project Engineer

APPROVED: Owen R. Lawter

OWEN R. LAWTER, Colonel, USAF
Chief, Intelligence and Reconnaissance Division

FOR THE COMMANDER: John P. Huss

JOHN P. HUSS
Acting Chief, Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (IRAA) Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return this copy. Retain or destroy.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: RADC-TR-80-365	2. GOVT ACCESSION NO: AD-A097657
3. RECIPIENT'S CATALOG NUMBER:	
4. TITLE (and Subtitle): CO-CHANNEL INTERFERENCE SEPARATION	5. TYPE OF REPORT & PERIOD COVERED: Final Technical Report. Sep 79 — Sep 80
	6. PERFORMING ORG. REPORT NUMBER: PAR-80-57
7. AUTHOR(s): Dr. Robert J. Dick	8. CONTRACT OR GRANT NUMBER(s): F30602-79-C-0278
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Pattern Analysis and Recognition Corporation, 228 Liberty Plaza, Rome NY 13440	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 31011G, 70550737
11. CONTROLLING OFFICE NAME AND ADDRESS: Rome Air Development Center (IRAA), Griffiss AFB NY 13441	12. REPORT DATE: December 1980
	13. NUMBER OF PAGES: 99
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Same	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION DOWNGRADING SCHEDULE: N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: Melvin G. Manor, Jr. (IRAA)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Co-channel interference
Multi-channel interference
Speech Processing
Communication interference
Computer programs
Software

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

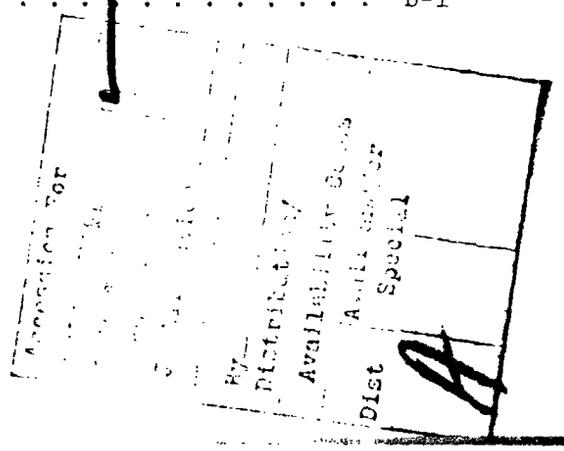
The objective of this effort was to investigate techniques to reduce the signal degradation due to co-channel and adjacent-channel voice-on-voice interference. Sorting and suppression techniques using linear prediction coefficients and an adaptive comb filter were investigated. Limited success was achieved. A procedure for estimating the frequency shift due to mistuning was developed.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

TABLE OF CONTENTS



LIST OF FIGURES

EVALUATION

This contract was in support of TPO 4E2, Technology Reconnaissance and Intelligence/Speech Processing. Communications are frequently degraded due to co-channel and adjacent-channel voice-on-voice interference. This effort investigated techniques for reducing the degradation. Additional work is programmed in this area.

Melvin G. Manor, Jr.

MELVIN G. MANOR, JR.
Project Engineer

SECTION 1
INTRODUCTION

This report presents the results of a project principally aimed at resolving multivoice data into separate voice signals. A secondary aim was to resolve noisy multivoice data where one or more voices may be distorted. Such a situation might result from adjacent radio channel interference due to transmitter or receiver nonlinearities.

The report is organized mainly by chronological order. Section 2 describes an attempt to identify to which of two speakers various sounds belong. The method investigated was identification by adaptive lattice filter coefficients. It was determined that this is not practical.

Section 3 describes an attempt to reduce the effect of a primary speaker masking a secondary speaker. The method used was speech dependent comb filtering. The section starts with a description of complex correlation, an algorithm used extensively throughout the successful phases of the project. It concludes with a description of comb filtering and its partial success.

Section 4 returns to adaptive lattice filtering, this time as a possible means of suppressing a primary voice. Listening tests showed that it did not separate voices.

Section 5 describes the sorting of multivoice sounds given that certain special cases are satisfied. Section 6 describes the combining of the work of Sections 3 and 5. The result is the final voice processing developed during the effort. Section 6 also describes an automatic way to estimate speech frequency shift. This is believed to be a new result. Section 7 gives conclusions, and recommendations for further work.

Appendix A describes the algorithm used to generate and correct speech frequency shifts. Finally, Appendix B describes the project software, and gives listings.

SECTION 2
STUDY OF LPC COEFFICIENTS

The first stage of the project was the study of lattice coefficients, also known as linear predictive coding (LPC) coefficients. Other terms for them are reflection coefficients and partial correlation (PARCOR) coefficients. The object was to determine if the coefficients could be used to assign various sounds during mixed speech to their respective speakers. It was concluded that this is not practical.

2.1. LATTICE FILTERING

The starting point for the first stage of the project was a paper by Markel et al [2] entitled, "Long-Term Feature Averaging for Speaker Recognition." In this paper, the authors report of an experiment in identifying speakers by pitch, gain, and reflection coefficients. They found the not surprising result that the longer the coefficients were averaged, the better they could be used to discriminate between speakers.

The authors digitized their data at 6.5 kHz, and applied pre-emphasis, before performing 10 stage lattice filtering. A digitizing rate of 6400 Hertz was selected for the present project. Pre-emphasis was applied by differencing the data ($1-z^{-1}$) before lattice filtering. The number of samples for reflection coefficient computation (the frame size) was 128 (20 milliseconds) for both the first stage of the present project and the work of Markel et al.

The method used in the present project for computing reflection coefficients was Makhoul's method F [1], the harmonic mean method, credited to Burg. Makhoul stated that he tended to prefer the use of the harmonic mean method because it minimizes a reasonable and well-defined error criterion. Figure 2-1 illustrates the flow of data through

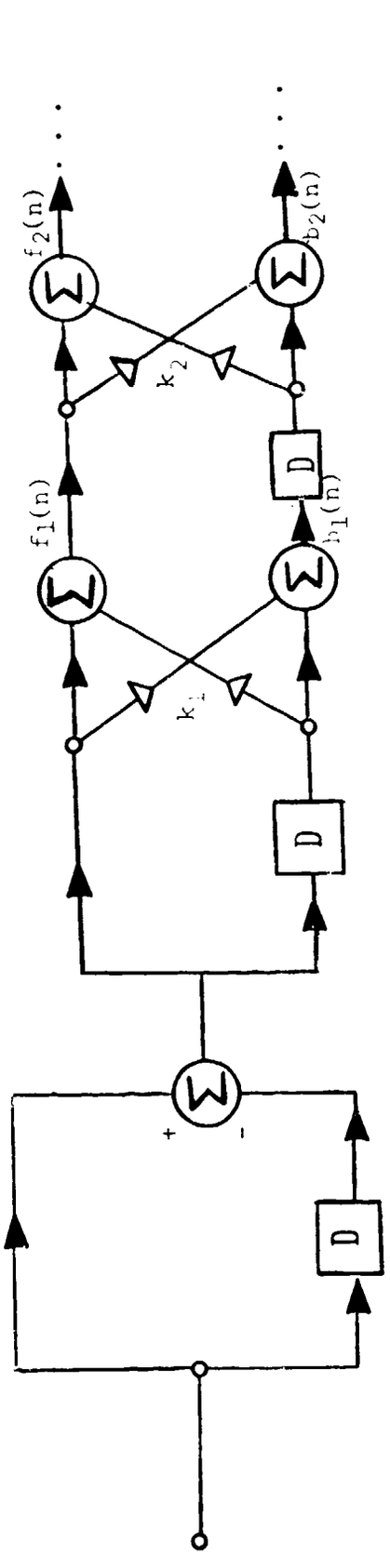


Figure 2-1 Lattice Analysis Filter

a lattice filter used for speech analysis. The summer at the left is for signal pre-emphasis. $f_1(n)$ and $f_2(n)$ are the first and second forward residuals at time n. $b_1(n)$ and $b_2(n)$ are the first and second backward residuals at time n. k_1 and k_2 are the first and second reflection coefficients. Coefficient k_s at stage s is computed as

$$k_s = -2(\sum_n f_{s-1}(n)b_{s-1}(n-1)/(\sum_n f^2_{s-1}(n) + \sum_n b^2_{s-1}(n-1))$$

The sum over n was taken over one frame interval, namely 128 points (20 milliseconds), for this stage of the project. The reflection coefficients are calculated in order. Once reflection coefficient k_s has been computed, $f_s(n)$ and $b_s(n)$ are computed for all n in the frame as

$$f_s(n) = f_{s-1}(n) + k_s b_{s-1}(n-1)$$

and

$$b_s(n) = k_s f_{s-1}(n) + b_{s-1}(n-1) \quad .$$

Then after f_s and b_s are computed k_{s+1} may be computed. As Makhoul showed, computing k_s as just given minimizes the sum of the squares of f_s and b_s.

Figure 2-2 shows a graphic output of a lattice filtering program applied to some single speaker data. The lines are organized in groups of three. There are five frames per line, and each line in each frame is normalized. The first line of each group shows the original waveform data. The second part of each group is a series of bar graphs, with each bar graph showing the 10 reflection coefficients for one frame. The third line of each group shows the forward residual after 10 stages of lattice filtering. This forward residual resembles an impulse train during voiced speech.

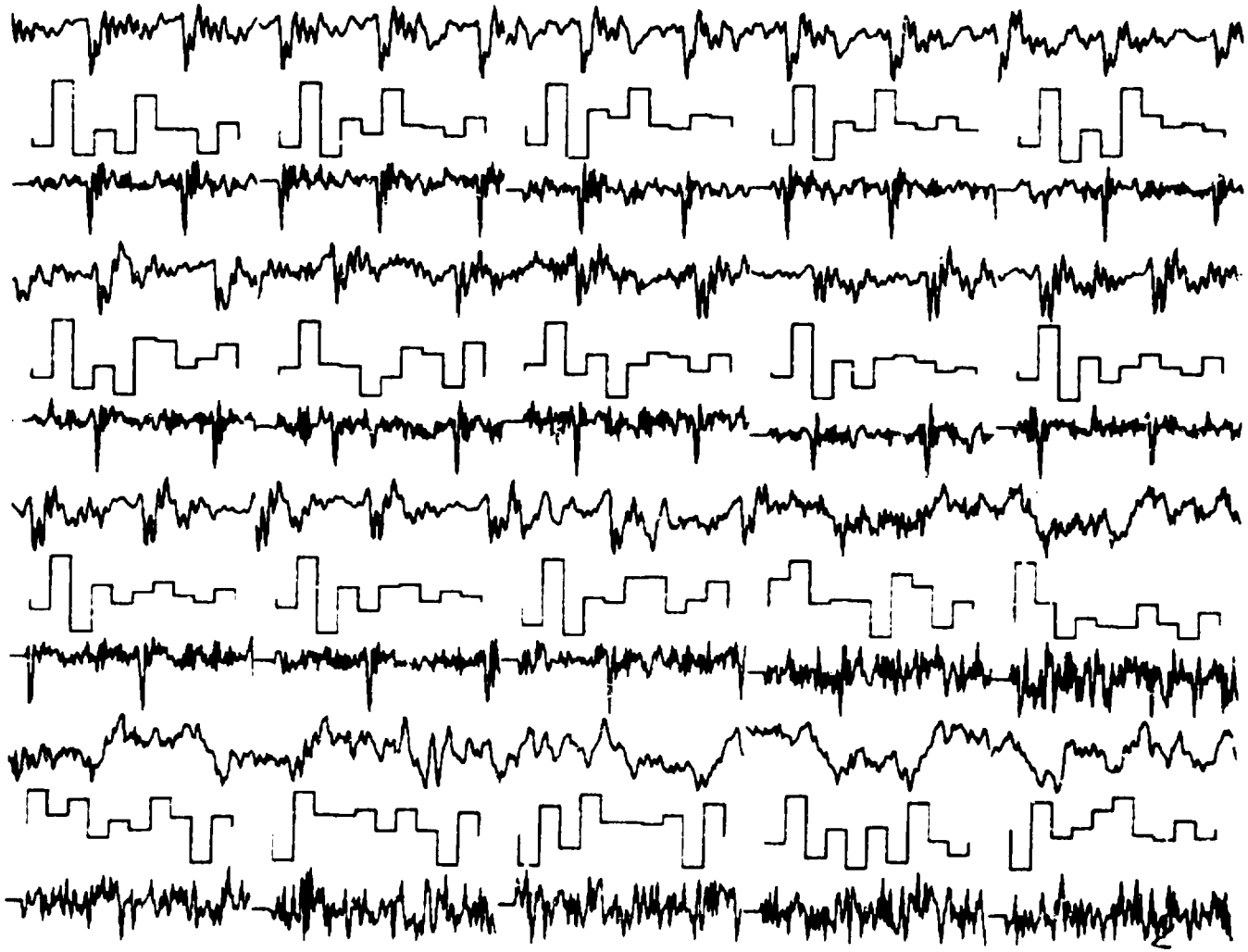


Figure 2-2 Result of Lattice Filtering

2.2. LATTICE FILTERING RESULTS

Four male speakers were recorded, each speaking the same set of four sentences, of five seconds duration each. This gave a total of 16 data samples. For each speaker and each sentence, a two second average was taken for each of the ten reflection coefficients. In computing each average, a weight was given to each frame proportional to the signal power in the frame. This was to reduce the weighting of reflection coefficients computed during silences or conconants.

The results were analyzed with the On-Line Pattern Analysis and Recognition System (OLPARS). It was found that coefficients 3 and 6 were best for discriminating between speakers. This is similar to the results of Markel et al, who found that coefficients 2 and 6 gave the best discrimination. The difference in result may be due to the small data base of the present effort. Figure 2-3 shows an OLPARS plot in the plane of reflection coefficients 3 and 6. Letters A through D represent the four speakers. The four data points for each speaker are connected by a polygon drawn by hand.

The OLPARS plot shows that in this case speakers A, B, and D can be distinguished, while speaker C data overlaps the data for both speakers A and D. This result is not encouraging, because two seconds of data is still much more than can be obtained from a single syllable. Markel et al determined that reflection coefficient scattering is inversely proportional to the cube root of the number of voiced samples averaged. Therefore, decreasing the averaging interval by a factor of 27 (from 2 seconds to 74 milliseconds) would increase the scatter for each speaker by a factor of three. This would cause the data for the four speakers to overlap substantially.

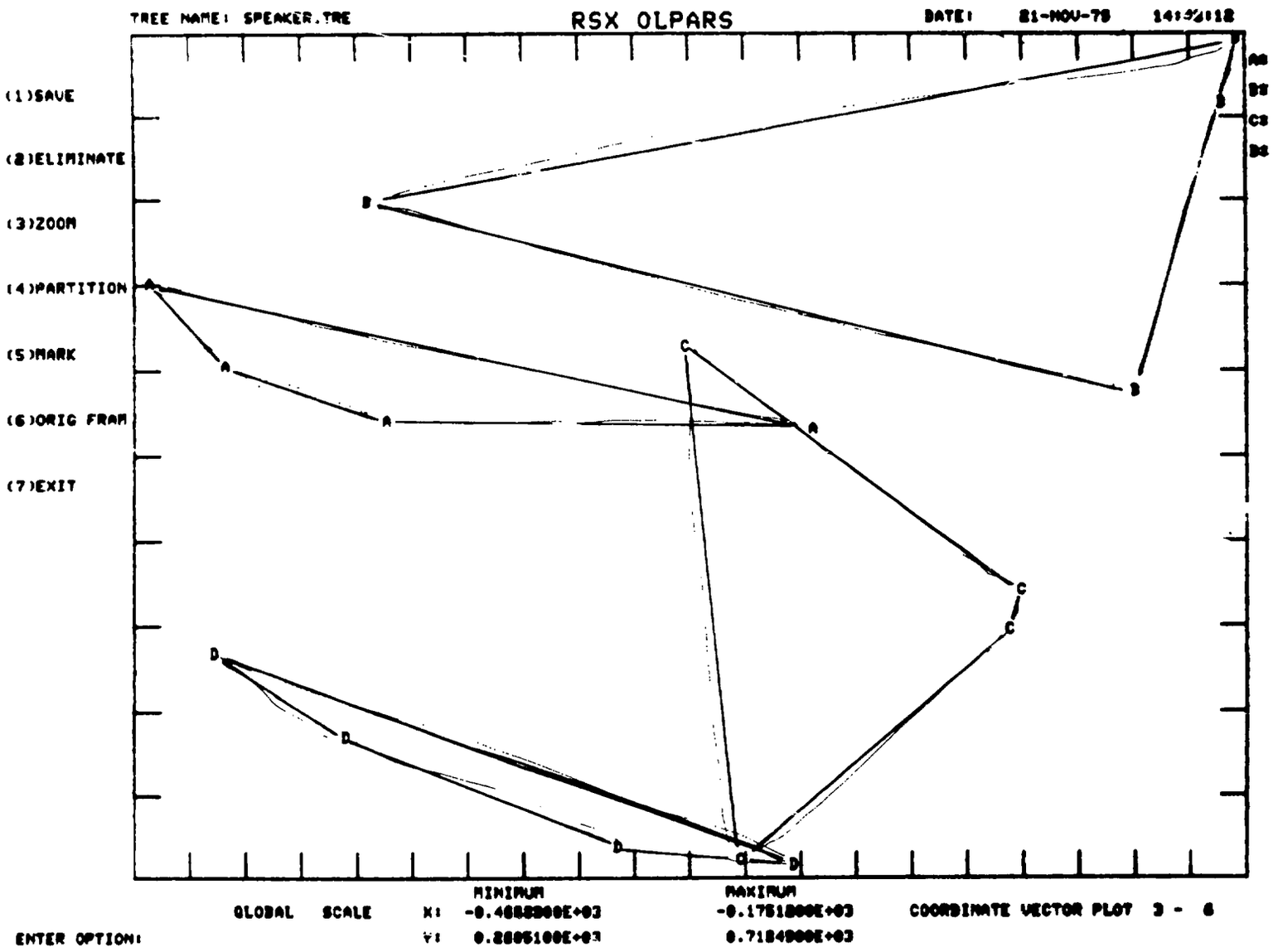


Figure 2-3 Result of OLPARS Processing

An additional problem is that reflection coefficients are sensitive to noise and to competing speaker signals. It was concluded that sorting syllables based on reflection coefficients would not be reliable. Additional work with LPC lattice filtering is discussed in Section 4.

SECTION 3
SPEECH DEPENDENT COMB FILTERING

3.1. COMPLEX CORRELATION

By complex correlation is meant an operation similar, but not identical, to computing an autocorrelation. An autocorrelation may be computed by taking an FFT, followed by taking the magnitude squared, followed by taking an inverse FFT. As the term will be used here, complex correlation starts with taking a tapered window of the data, namely the Hanning (raised cosine) window. This is followed by taking an FFT, and then taking the magnitude squared. The result is the power spectrum. This corresponds so far to the operations in calculating an autocorrelation. Next, however, for taking a complex correlation, the negative frequency portion of the transform is zeroed. For the discrete Fourier transform of N points, this is equivalent to zeroing the values at N/2, N/2+1, ..., N-1. No information is lost by this operation, because the power spectrum of a real signal is symmetric about zero frequency. The discrete power spectrum of a pure-real sampled time signal is also symmetric about half the sampling frequency. This means that the discrete power spectrum values at N/2, N/2+1, ..., N-1 are the mirror images of the values at 0, 1, ..., N/2-1. Thus when half the power spectrum is zeroed only redundant information is lost.

Next in computing the complex correlation, the square roots of the remaining points are taken. This operation is done to reduce the ratio of peaks in the data. For example, a power spectrum with one peak 25 times as high as another will have this ratio reduced to 5 after the square root is taken. Since the complex correlation will be used to estimate pitch, it is important that it be influenced by several pitch harmonics, and not just by a single dominant power spectrum peak.

Finally, the inverse FFT is taken. The result is a complex function, because it is the inverse transform of a function with only positive frequency components. Hence the term complex correlation. One way of viewing the complex correlation is that each point in it represents a coefficient in the Fourier series which can reproduce the processed power spectrum. For S(0), S(1), ..., S(N-1) denoting the processed power spectrum, we have that the correlation C(0), C(1), ..., C(N-1) is given by

$$C(n) = \frac{1}{N} \sum_{k=0}^{N-1} S(k) \exp(j2\pi kn/N)$$

for n = 0, 1, ..., N-1.

See [3] page 89. Then S(k) is represented by the Fourier series with coefficients C(n), that is by

$$S(k) = \sum_{n=0}^{N-1} C(n) \exp(-j2\pi kn/N)$$

However, S(k) is pure real, so we may rewrite this as

$$S(k) = \sum_{n=0}^{N-1} \mathrm{Re}(C(n)) \cos(2\pi kn/N)$$

$$+ \sum_{n=0}^{N-1} \mathrm{Im}(C(n)) \sin(2\pi kn/N).$$

3.2. COMPLEX CORRELATION APPLIED TO SPEECH DATA

Speech data during voiced speech is characterized by the presence of multiple harmonics of the pitch. During unvoiced speech, the data is characterized as filtered broadband noise. However, the unvoiced portions are relatively less important for two reasons. First, unvoiced speech has considerably less power than does voiced speech. Thus in

noisy data, it is to a considerable extent masked by the noise. Second, listening tests have shown that the human ear is relatively insensitive to distortion in the reproduction of unvoiced sounds. For these reasons, the speech processing under the present effort was limited to treating all data as if it consisted of voiced speech. Listening tests of the results confirm that the resulting suboptimal processing of unvoiced sounds is hardly noticeable. Indeed, it seems likely that attempting to make voiced/unvoiced decisions on noisy multitalker data would introduce more distortion (due to decision errors) than it would eliminate.

Figure 3-1 illustrates the square root of power spectrum versus time for a single talker. This is for microphone speech (as opposed to SSB radio reception) with only ambient room noise in the background. Each line represents a frequency range of 0 to 3200 Hertz, and time advances by 40 milliseconds from one line to the next. Many lines show the comb-like regular succession of peaks characteristic of voiced speech.

Figure 3-2 illustrates the magnitude of a complex correlation versus time for a single talker. This is also for microphone speech. This plot was produced by the program CCORPL, listed in Appendix B. Each horizontal line represents a time range of 0 to 20 milliseconds. Time advances by 20 milliseconds from one line to the next. The curve to the left of the horizontal lines shows speech amplitude versus time. Correlation peaks are visible on most of the horizontal lines, showing pitch periods of 8 to 10 milliseconds. The speaker's pitch is therefore in the range of 100 to 125 Hertz. Several portions of time where the speaker's pitch is not evident also have dips in speech amplitude. Absence of correlation peaks showing speaker pitch indicates silence, or non-voiced speech. The complex correlation provides an automated way of determining speaker pitch. It may also be used to determine the best fit of a comb filter to the data. This will be further discussed in the next subsection.

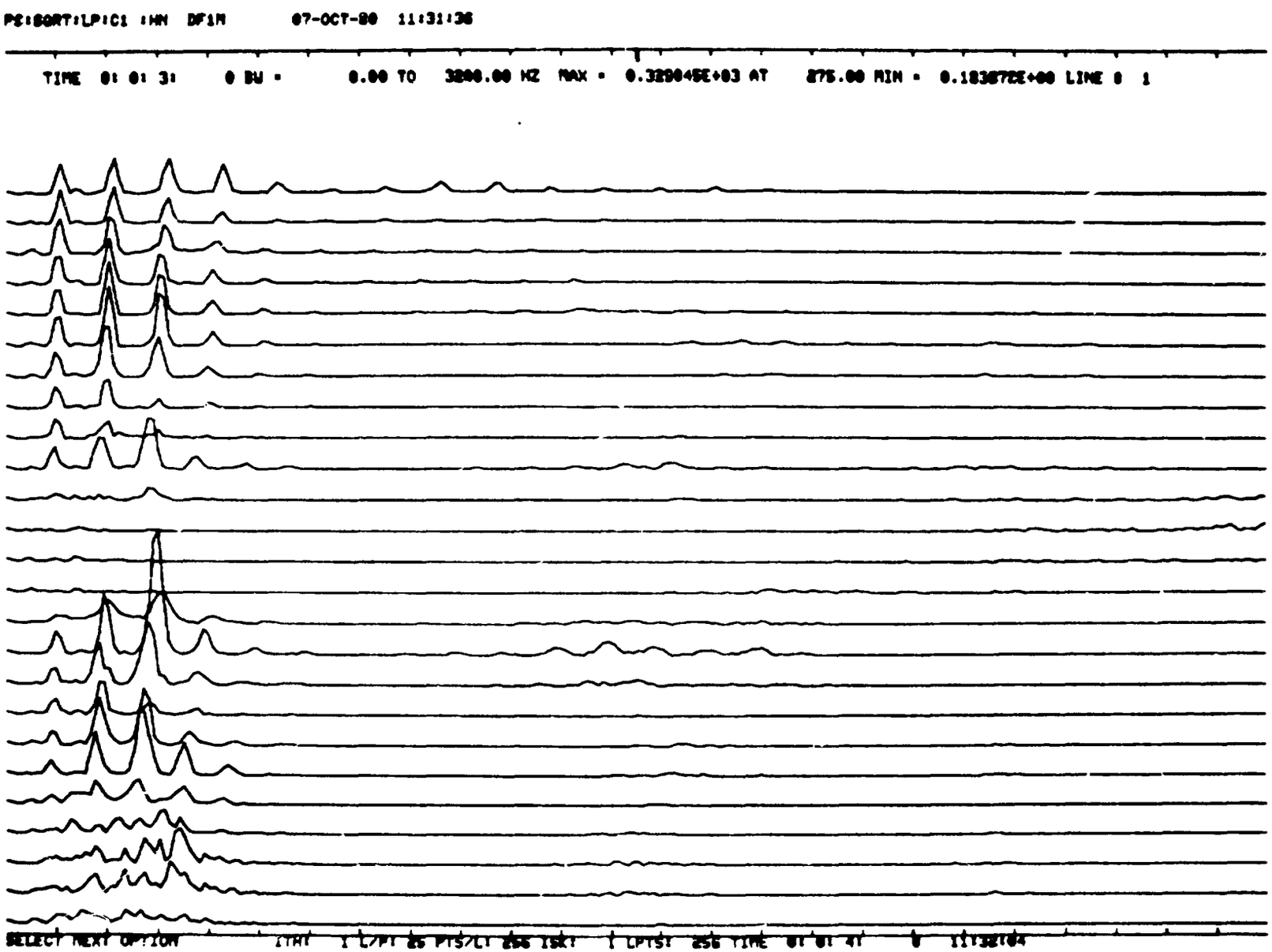


Figure 3-1 Square Roots of Speech Power Spectra

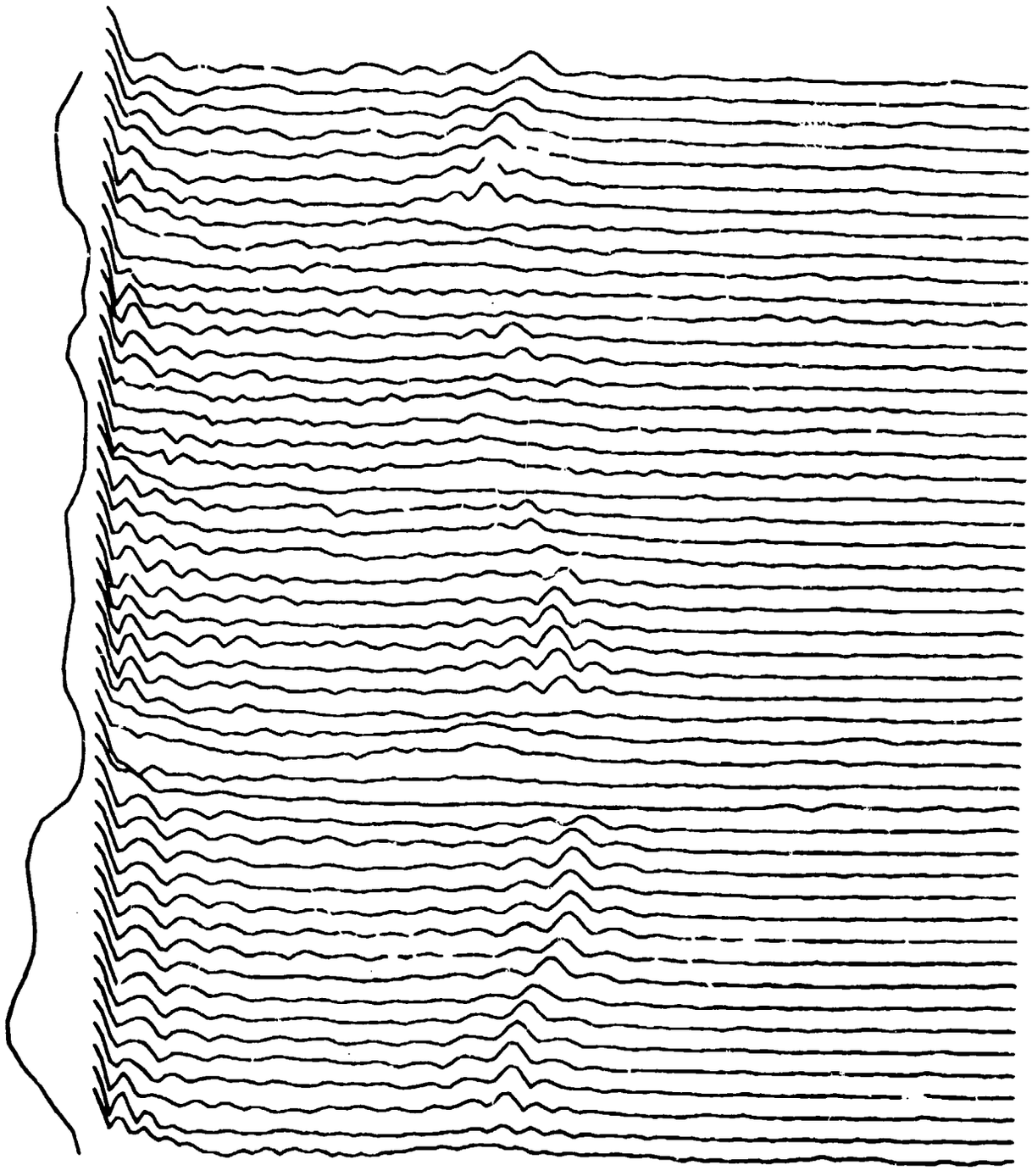


Figure 3-2 Magnitudes of Speech Complex Correlations

3.3. PRINCIPLES OF SPEECH DEPENDENT COMB FILTERING

By speech dependent comb filtering is meant applying a filter to the data with the filter itself dependent on the data. In particular, the object is to enhance or suppress the voiced speech of one speaker. As Figure 3-1 illustrates, voiced speech has a comb-like structure in the frequency domain. To enhance such data, a filter may be used which passes the speech peaks and attenuates the valleys. To suppress the speech, a filter may be used which attenuates the peaks while passing the valleys. The purpose of suppressing the speech is to let the speech of a second speaker come through with comparatively little attenuation versus the primary speaker.

From Section 3.1, we have that for S(k) the one-sided square root of the power spectrum and C(n) the complex correlation

$$C(n) = \frac{1}{N} \sum_{k=0}^{N-1} S(k) \exp(j2\pi kn/N)$$

We wish to find the pitch period T and frequency displacement d such that the comb function F(k) best fits S(k) where

$$F(k) = 0.5 + 0.5 \cos(2\pi Tk + d)$$

That is, we wish to find the T and d which maximize the dot product of F(k) and S(k):

$$F \cdot S = \sum_{k=0}^{N-1} 0.5[1 + \mathrm{Re}(\exp(j2\pi Tk)\exp(jd))]S(k)$$

$$= 0.5NC(0) + 0.5N\mathrm{Re}(C(T)\exp(jd))$$

The maximizing T may not be an integer. We will deal with this problem shortly. Given T and a complex C(T), then the maximizing d is any value which makes C(T) exp(jd) pure real. It follows that

$$\max_d F \cdot S = 0.5N(C(0) + |C(T)|).$$

Therefore the T which gives the best comb fit to the data is the T which gives the maximum value of the magnitude of the complex correlation. Of course, we must rule out very small values of T, because the complex correlation C(n) has its overall maximum value at n=0. We are interested only in the maximum value over the range of possible pitch periods.

The d which gives the best fit is such that $\exp(jd) = C(T)^*/|C(T)|$, where * denotes the complex conjugate. That is,

$$\cos d = \text{Re } C(T)/|C(T)| \text{ and}$$
$$\sin d = -\text{Im } C(T)/|C(T)|$$

It follows that

$$d = \arctan(-\text{Im } C(T)/\text{Re } C(T)).$$

Care must be taken in evaluating the arctan to put d into the proper quadrant. In Fortran, this may be done by using the ATAN2 function.

Now we consider the choice of a non-integer T. The first method used to select the best T was to first find the maximizing integer T_i, and then take the centroid (center of gravity) of the three points $C(T_i - 1)$, $C(T_i)$ and $C(T_i + 1)$. This gave the centroid value T_c as

$$T_c = \frac{(T_i - 1)C(T_i - 1) + T_iC(T_i) + (T_i + 1)C(T_i + 1)}{C(T_i - 1) + C(T_i) + C(T_i + 1)} .$$

Then

$$T_c - T_i = \frac{C(T_i + 1) - C(T_i - 1)}{C(T_i - 1) + C(T_i) + C(T_i + 1)} .$$

The method used to find the best d given a non-integer T was quadratic interpolation. Three points determine a quadratic curve, and the three points used were the best integer T (denoted T_i), along with $T_i - 1$ and $T_i + 1$. The real and imaginary parts of C(T) were interpolated separately.

For G(k) a function of integer valued k, we may fit to it the quadratic function $F(x) = G(0) + C_1 x + C_2 x^2$ where

$$C_1 = 0.5(G(1) - G(-1))$$

$$C_2 = 0.5(G(-1) + G(1)) - G(0)$$

The reader may verify that F(x) = G(x) for x = -1, 0, 1. Then for T_i the best integer T, and T_b the best (possibly non-integer) T, the real part of $C(T_b)$ was taken to be

$$\text{Re } C(T_b) = C(T_i) + (T_b - T_i)(R_1 + (T_b - T_i)R_2)$$

where

$$R_1 = 0.5 \text{ Re}(C(T_i + 1) - C(T_i - 1))$$

$$R_2 = \text{Re}[0.5(C(T_i - 1) + C(T_i + 1)) - C(T_i)] .$$

The imaginary part of $C(T_b)$ was computed similarly. These methods of interpolation were used through the time that the first few automatic mistuning estimates were made (see Section 6.2). It was found then that the mistuning estimates were consistently about -5 Hertz for microphone speech, which had never been mistuned. A revision was made to the

method of computing the optimal pitch period T. The revised idea was to choose the T which maximized the quadratic interpolation of $|C(n)|^2$ fitted to the three points $|C(T_i - 1)|^2$, $|C(T_i)|^2$, and $|C(T_i + 1)|^2$. This T_o may be determined as follows. For $F(x) = G(0) + C_1x + C_2x^2$ we have

$$F'(x) = C_1 + 2C_2x$$

$$F''(x) = 2C_2$$

so F(x) has its maximum at $x = -C_1/2C_2$, provided that C_2 is negative. For the quadratic interpolation of $|C(n)|^2$ we have

$$x = T_o - T_i$$

$$C_1 = 0.5(|C(T_i + 1)|^2 - |C(T_i - 1)|^2)$$

$$C_2 = 0.5(|C(T_i - 1)|^2 + |C(T_i + 1)|^2) - |C(T_i)|^2 \quad .$$

Therefore we choose

$$T_o = T_i - C_1/2C_2$$

provided that $|C(T_i)|^2 > 0.5(|C(T_i - 1)|^2 + |C(T_i + 1)|^2)$. This last condition is true any time it is true that $|C(T_i)|^2$ is greater than both $|C(T_i - 1)|^2$ and $|C(T_i + 1)|^2$. Using the quadratic interpolation value for T_o was found to correct the mistuning estimates.

3.4. VARYING COMB TOOTH THICKNESS

At one point in the project, it was thought that varying the thickness of the comb teeth would aid in speech enhancement or suppression. The reasoning for this was that when speech was suppressed, typically

the resulting signal still had its spectral peaks where the original signal had them. This was believed to be due to frequency quantizing preventing the voice suppression filter (referred to as a rake filter) from ever being exactly zero. Figure 3-3 illustrates the phenomenon.

Possibly widening the valleys of the rake filter would suppress the voiced speech even more effectively, and as a result, let some secondary speech come through better. Similarly, if a secondary speech peak coincided with a rake filter peak, then possibly widening the rake filter peak (its "tooth") would allow the secondary speech signal to come through better.

The method used to make peaks and valleys wider or narrower was to warp the frequency axis when applying the comb. That is, instead of the comb

$$F(k) = 0.5 + 0.5 \cos(2\pi Tk + d)$$

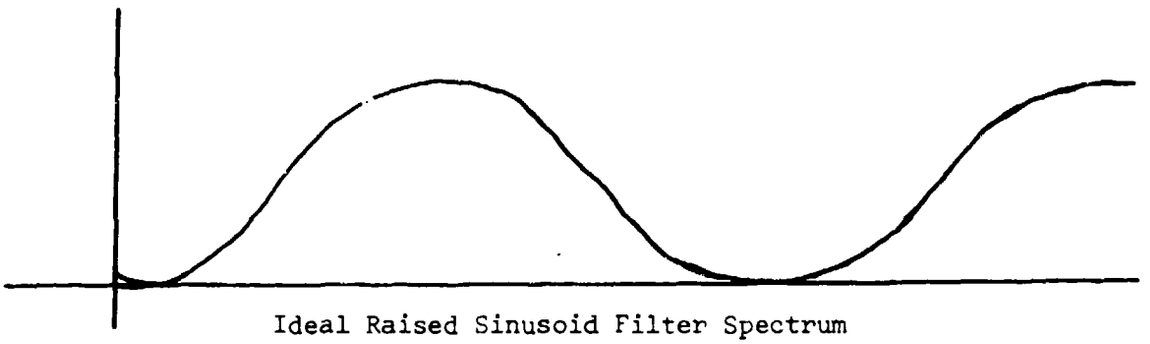
the frequency warped comb F(z(k)) was used:

$$F(z(k)) = 0.5 + 0.5 \cos(2\pi Tz(k) + d).$$

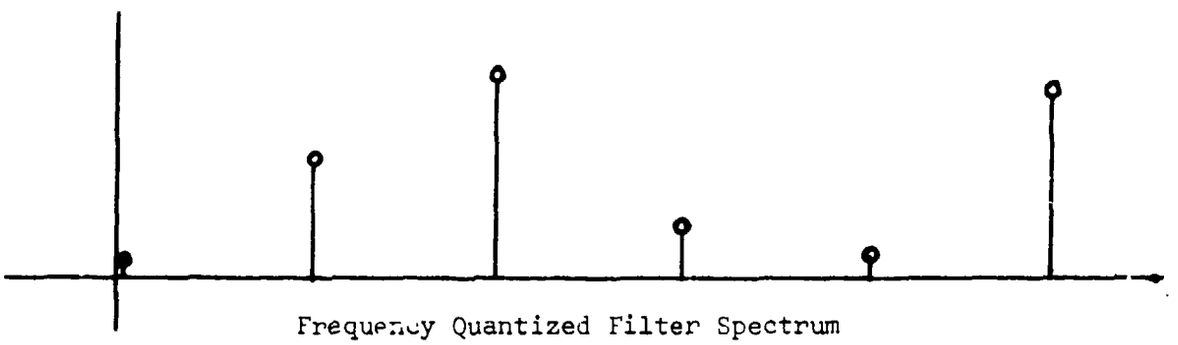
Warped frequency z(k) was determined as follows: For each frequency point k, the points x_0 and x_1 are such that x_0 is the location of the comb peak or valley nearest to but preceding k. x_1 is the peak or valley nearest to but following k. Then if x_0 is a peak location, x_1 is a valley location, and vice versa. Then for S(n) the square root of the power spectrum, z(k), is computed to be

$$z(k) = k + K(S(x_0) - S(x_1))(k - x_0)(k - x_1)$$

where K is such that



Ideal Raised Sinusoid Filter Spectrum



Frequency Quantized Filter Spectrum

Figure 3-3 Effect of Quantizing on Filter Spectrum

$$1/K = (S(x_0) + S(x_1))(x_1 - x_0)$$

For non-integer x_0 and x_1 linear interpolation is used to compute $S(x_0)$ and $S(x_1)$ respectively. Then $z(k) = k$ for $k = x_0$ or x_1.
Also

$$z(\frac{x_0 + x_1}{2}) = \frac{x_0 + x_1}{2} + \frac{(S(x_1) - S(x_0))(x_1 - x_0)}{4(S(x_0) + S(x_1))}$$

So $z(\frac{x_0 + x_1}{2}) > \frac{x_0 + x_1}{2}$ if and only if $S(x_1) > S(x_0)$.

Thus if a power spectrum peak matches either a comb peak or valley, then that comb peak or valley is widened to include more of the power spectrum peak. The coefficient K in the warping formula is the maximum value consistent with not warping points in the interval x_0 to x_1 so much that they are moved outside the interval. This is equivalent to the condition that $z'(x_0) \geq 0$, and $z'(x_1) \geq 0$. We have

$$z'(k) = 1 + K(S(x_0) - S(x_1))(k - x_0 + k - x_1)$$

Now $$-1 \leq \frac{S(x_0) - S(x_1)}{S(x_0) + S(x_1)} \leq 1$$

so

$$z'(x_0) \geq 1 + (\frac{1}{x_1 - x_0})(x_0 - x_1) = 0$$

and

$$z'(x_1) \geq 1 - (\frac{1}{x_1 - x_0})(x_1 - x_0) = 0.$$

3.5. COMB VOICE PROCESSOR AND RESULTS

Figure 3-4 gives a block diagram of the data flow within the comb voice processor. The input data is digitized at 6400 Hertz. It is divided into 512 point (80 millisecond) intervals, with each interval

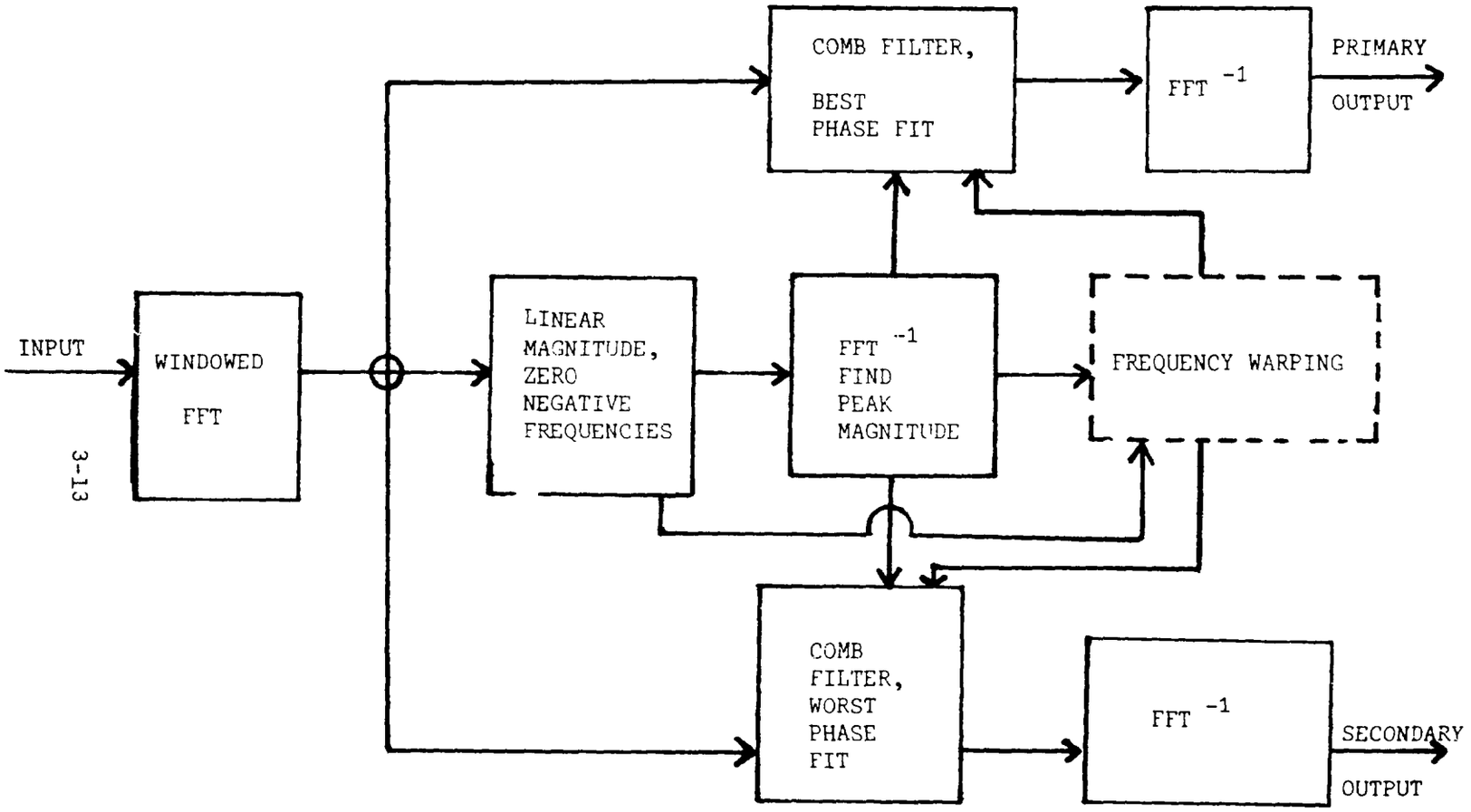


Figure 3-4 Comb Voice Processor Data Flow

advanced 256 points from the previous one. That is, the intervals have 50 percent overlap. Each window is given the Hanning (raised cosine) weighting, and a fast Fourier transform (FFT) is taken. This transform is saved for further use. The square root of the power spectrum is computed from the FFT with negative frequency portions zeroed. Next an inverse FFT is taken of the root power spectrum. This is the complex correlation of the signal, as described in Section 3.1. Next the magnitude peak is found for the complex correlation, and the phase at the peak is computed. The search is performed over points 25 to 80 of the complex correlation, corresponding to speaker pitches of 81 to 267 Hertz. The location of the correlation peak and the phase at the peak allow determination of the best comb filter fit to the data, as described in Section 3.3.

At this point, frequency warping may be applied. A comparison is made of the linear magnitude power spectrum versus the locations of the comb filter peaks and valleys. The frequency axis is warped accordingly, as described in Section 3.4. The same frequency warping is used both for speech enhancement and suppression. This is because the enhancement comb peaks line up with the suppression comb valleys, and the frequency warping does not distinguish between peaks and valleys.

The FFT of the original signal is multiplied by the best fit raised sinusoid (possibly with frequency warping) and the inverse FFT is taken to give the primary output. Multiplication in the frequency domain is equivalent to convolution in the time domain, so the effect is a linear filtering of the signal. A secondary output is produced by multiplying the signal FFT by a raised sinusoid (possibly with frequency warping) with the same spacing between the teeth as for the primary output. However, for the secondary output, the comb is positioned so that overall its teeth line up against valleys in the signal.

The comb voice processor by itself was a partial success, as Figures 3-5 through 3-9 illustrate. Each of these figures shows the evolution of a 0 to 3200 Hertz spectrum versus time. Each line is derived from a 40 millisecond (256 point) Hanning, or raised cosine, time window. There is a 10 millisecond advance between lines. Figures 3-5 and 3-6 show spectra for speakers A and B, respectively. Figure 3-7 shows spectra for the speech of speaker A mixed at 6 dB above (at four times the power of) the speech of speaker B. The B speech shows up mainly as a perturbation to the A speech. Figure 3-8 shows spectra for the mixed speech subjected to the suppression algorithm without frequency warping. Figure 3-9 is for suppression with frequency warping. The result of suppression without frequency warping shows considerably more evidence of speaker B than does the original mixed speech. Comb filtering is therefore at least a partial success. The result of suppression with frequency warping is inferior to the result of suppression without frequency warping. For this reason, frequency warping was not used in the later stages of the project. Incidentally, the voice processing whose results are shown here used the quadratic interpolation method of calculating speaker pitch.

Also tried was saturating (clipping) the raised sinusoid frequency response so that it is zero at 25% of the frequencies, one at 25% of the frequencies, and transiting between zero and one for 50% of the frequencies. There seemed to be a slight but not significant degradation versus the raised sinusoid comb. The saturated comb was not used in the later stages of the project.

The initial hope for the comb voice processor was that it would be enough by itself to make intelligible a secondary voice, which prior to processing had been masked by a primary voice. This turned out not to be the case. The probable cause of the failure is that there are significant amounts of time, perhaps 20 to 30 percent, when the primary speaker is not producing sound, but the secondary speaker is. During

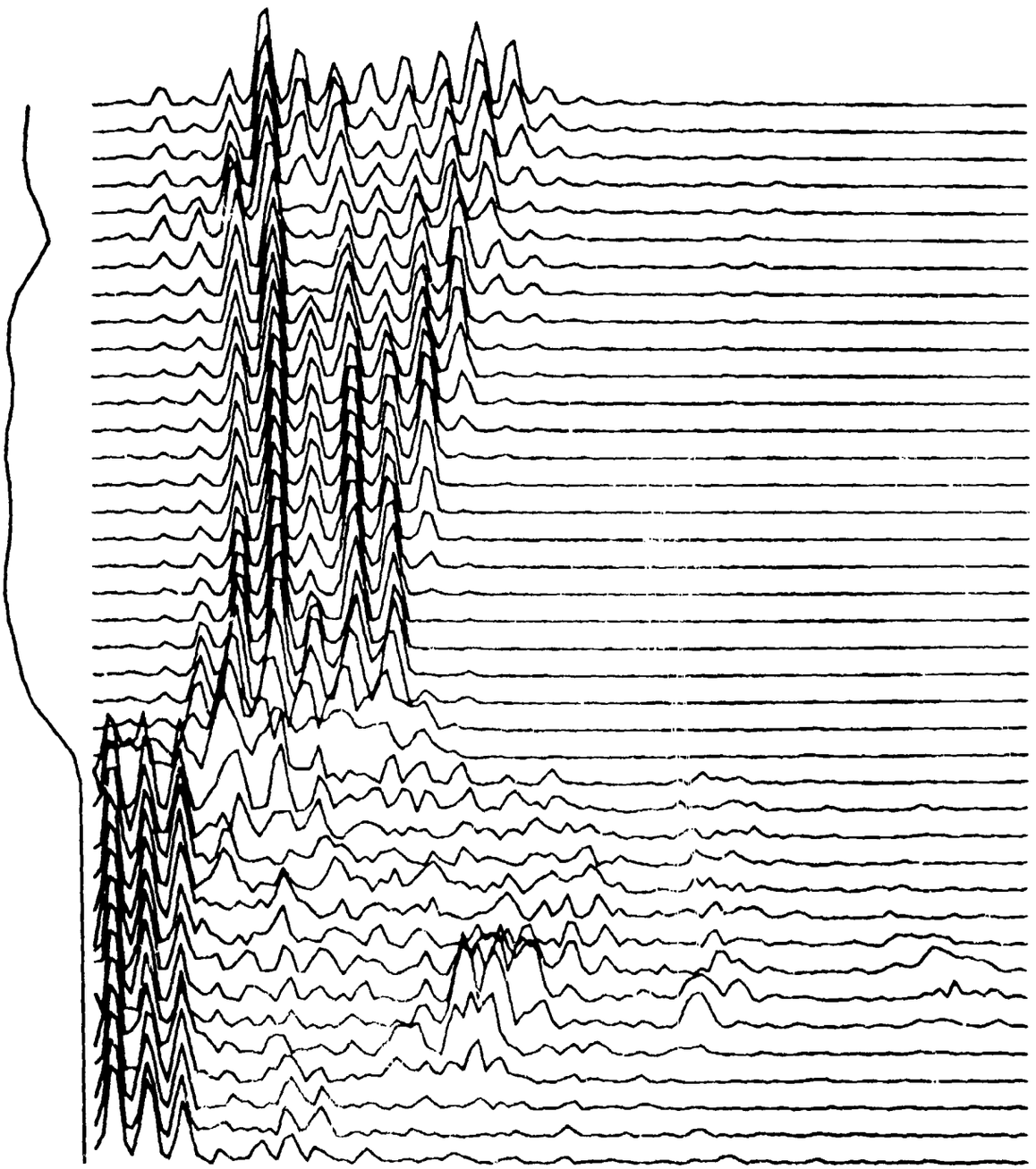


Figure 3-5 Speaker A Spectra

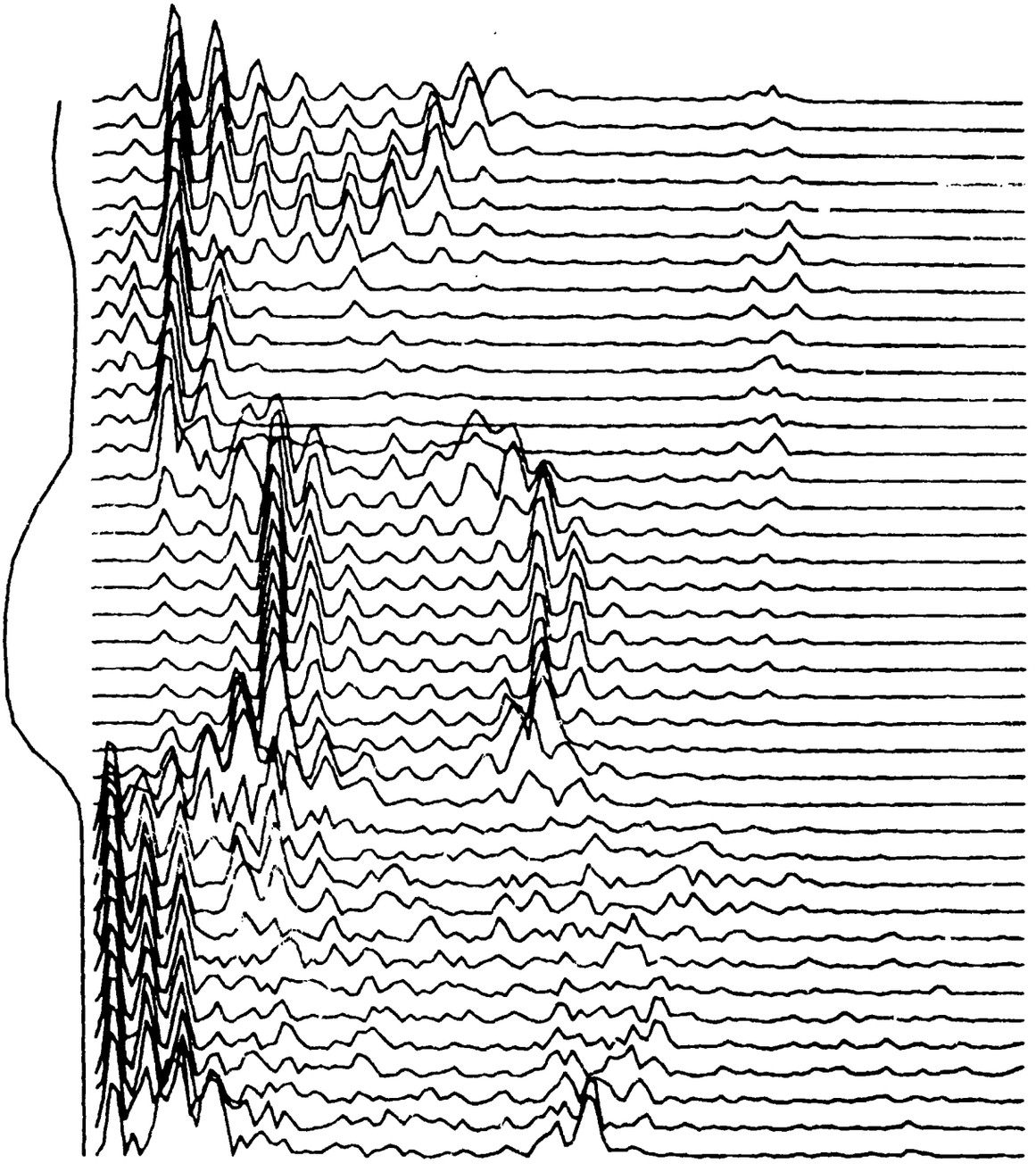


Figure 3-6 Speaker B Spectra

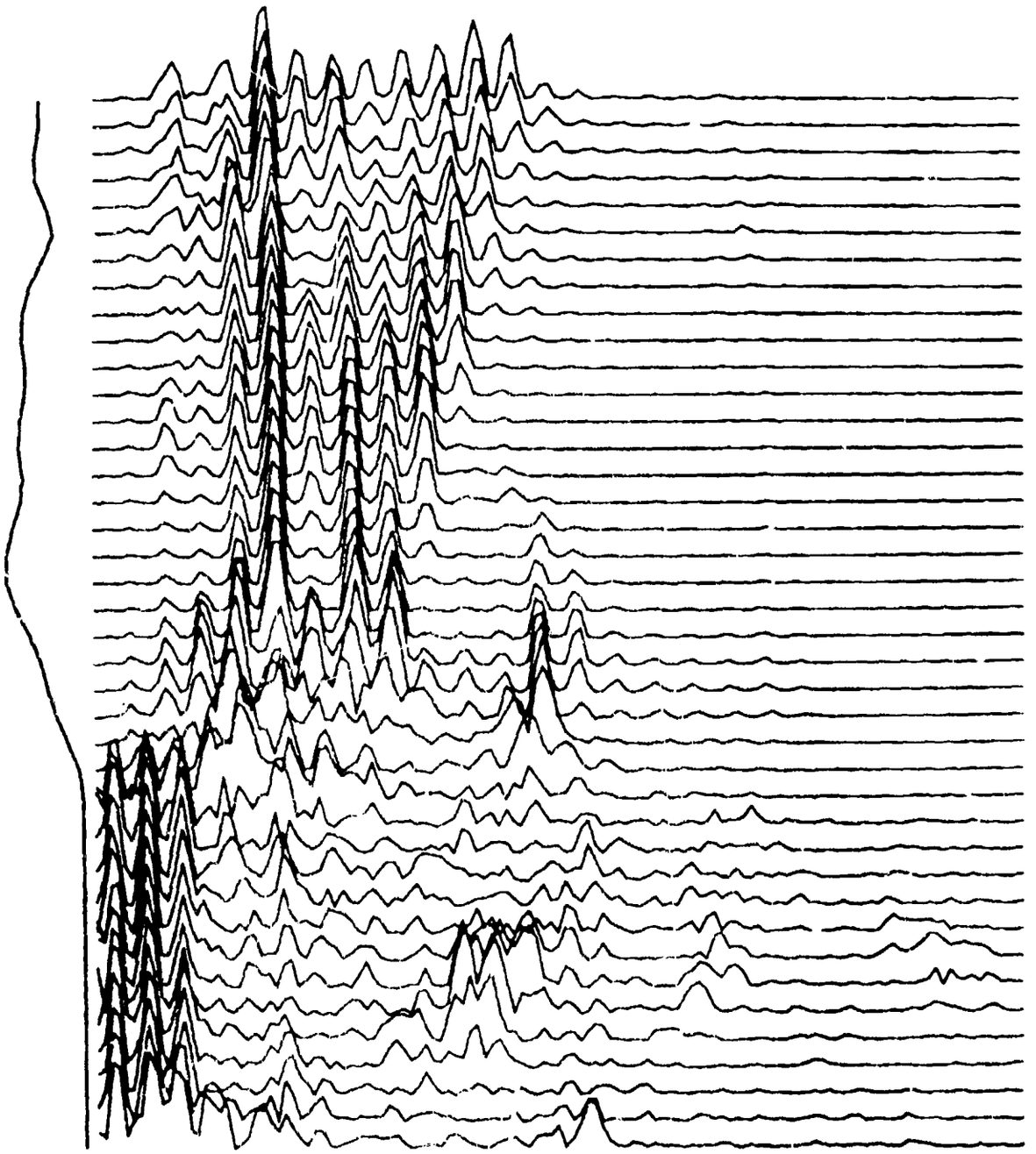


Figure 3-7 Mixed Speech Spectra

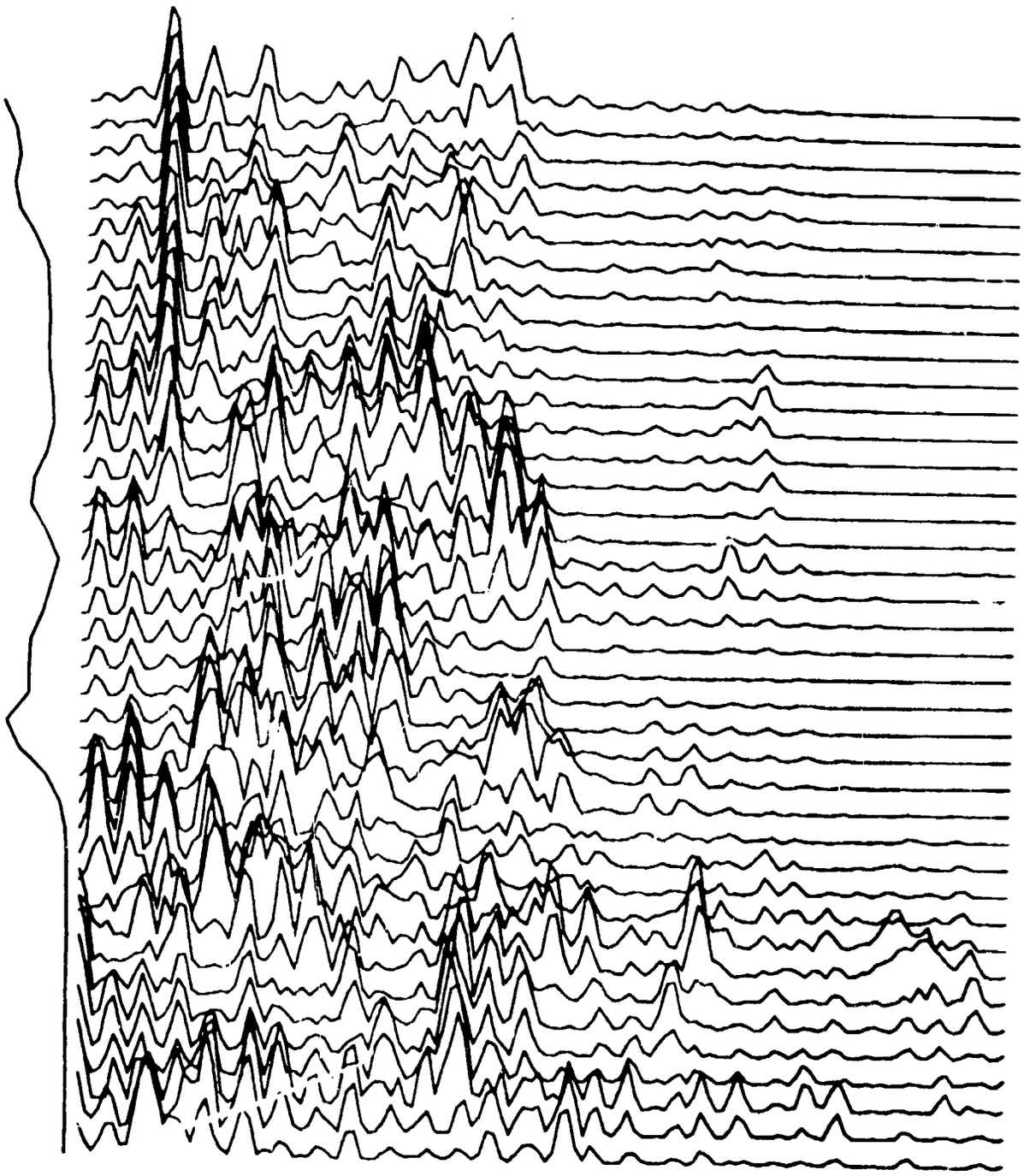


Figure 3-8 Spectra of Mixed Speech After Suppression

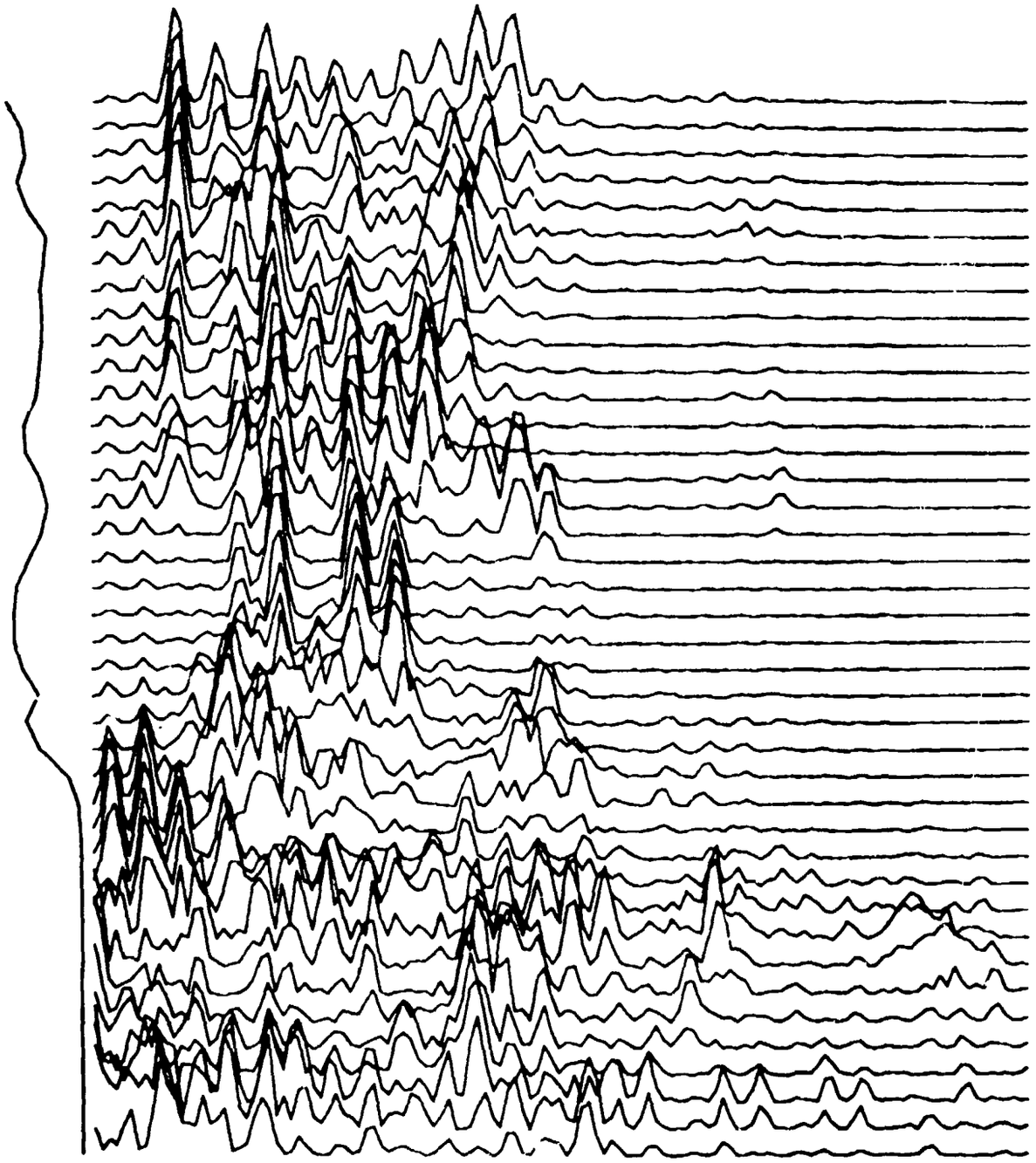


Figure 3-9 Spectra After Suppression with Frequency Warping

these times a pure suppression filter will lock on to the secondary voice and suppress it. What is needed is a way for the signal processing algorithm to distinguish intervals when one voice is dominant, versus intervals when the other voice is dominant. Two methods for doing this for special cases are discussed in Sections 5 and 6.

SECTION 4

LPC ANALYSIS AND RECONSTRUCTION

Section 2 discussed LPC lattice filtering with the object of assigning syllables in mixed speech data to their respective speakers. Here we discuss the use of lattice filtering as a stage of speech processing. In this portion of the project, lattice filtering was followed by thresholding the forward residual, followed by waveform reconstruction. The hope was that the thresholding would pass a dominant voice while rejecting a secondary voice. The reconstructed signal might then be subtracted from the original data, leaving the secondary voice. It turned out that the procedure did not separate the voices.

4.1. SPEECH RECONSTRUCTION BY LATTICE FILTERING

Figure 2-1 illustrates the initial stages of a lattice analysis filter. As described in Section 2.1, the forward residual f is related to the backward residual b by

$$f_s(n) = f_{s-1}(n) + k_s b_{s-1}(n-1)$$

and

$$b_s(n) = k_s f_{s-1}(n) + b_{s-1}(n-1).$$

We may solve these equations for $f_{s-1}(n)$ and $b_s(n)$, giving

$$f_{s-1}(n) = f_s(n) - k_s b_{s-1}(n-1)$$

and

$$b_s(n) = k_s f_s(n) + (1-k_s^2)\, b_{s-1}(n-1).$$

Figure 4-1 illustrates how these equations may be implemented. The processing of Figure 4-1 is the inverse of the processing of Figure 2-1.

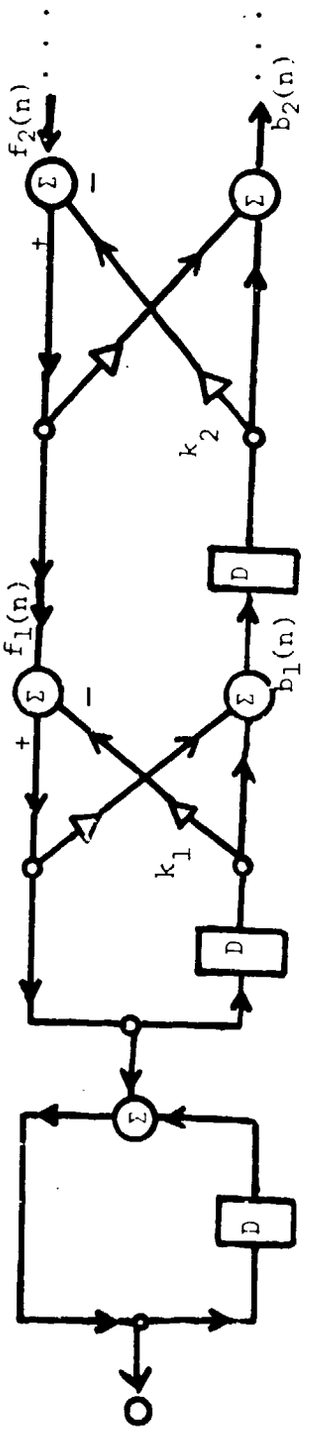


Figure 4-1 Lattice Synthesis Filter

The summer circuit on the left in 4-1 is the reciprocal of the differencing circuit on the left in Figure 2-1. The speech synthesis circuit of Figure 4-1 requires only the forward residual from the analysis filter of Figure 2-1.

In the analysis filter, the s-th residuals $f_s(n)$ and $b_s(n)$ are computed for fixed s and for all times n before the computation of $f_{s+1}(n)$ and $b_{s+1}(n)$. By contrast, in the synthesis filter, $f_s(n)$ and $b_s(n)$ are computed for fixed time n and all stages s before the computation of $f_s(n+1)$ and $b_s(n+1)$. The synthesis filter memories are assumed to be filled with zeroes initially.

4.2. LPC VOICE PROCESSOR AND RESULTS

The voice analysis and reconstruction described in Section 4.1 was programmed. Twenty stages of lattice filtering were used. The theory was that ten stages are suitable for a one voice signal, and a two voice signal may be twice as complex as a one voice signal. Data segments of 148 points (23.1 milliseconds) were used, with 20 point (3.1 millisecond) overlaps between segments, to allow for lattice filter startup effects. The analysis filtering of each segment resulted in a 20 point startup and a 128 point forward residual.

The root mean square (rms) value of the forward residual was calculated. Then the magnitude of each point in the residual was compared to a threshold consisting of the rms value times a constant. Primary and secondary residuals were then computed. The primary residual contained all residual values whose magnitudes exceeded the threshold, and was filled with zeroes elsewhere. The secondary residual contained all residual values whose magnitudes fell below the threshold, and was filled with zeroes elsewhere.

Figure 4-2 shows a result from the processing of a two-voice signal with this algorithm. In this case, the threshold for the magnitude of the residual was set at one times the rms value. The lines of the page are organized into groups of four. There are five frames per line, and each line in each frame is normalized. The first line of each group shows the original waveform data. The second line of each group shows the forward residual after 20 stages of lattice filtering. The third line shows the primary residual after thresholding. The fourth and last line of each group shows the primary output, that is, the output synthesized from the primary residual.

A program was written which produced primary and secondary output files. This program was named VCRFSP and is listed in Appendix B. The program was applied to mixed speaker data. Listening tests on the results showed that the primary output generally contained both voices. The secondary output contained both voices, but sounded as if both talkers were whispering. There is a logical explanation for this phenomenon.

A person speaking voiced speech produces an impulse train at his vocal cords, and this impulse train acoustically excites his vocal tract. By contrast, a whisperer does not produce impulses at his vocal cords. Instead he excites his vocal tract with the white noise generated by the rush of air through his throat. Of course, a voicing person also excites his vocal tract with white noise, but the effect is masked by the higher amplitude excitation of the impulse train. Lattice filtering in effect undoes the acoustic filtering of the vocal tract, resulting in a forward residual representing the excitation of the vocal tract. Then removing the impulses from this excitation will remove the voiced portion of the signal. Reconstruction will then result in making audible the whispering portion of a voicing speaker's sound. Unfortunately, this does not help separate voices.

Figure 4-2 Result of Voice Analysis and Reconstruction

A possible further course of research along LPC lines remains. It is to try to sort out mixed impulse trains in the forward residual, assigning one series of impulses to one speaker, and another series to another speaker. This line of research was not pursued for three reasons. First, sorting impulses would be difficult because the impulses are not always clearly distinct from the low level noise. Second, even if the impulses could be isolated, and impulse trains separated, there would be the problem of identifying which speaker produced which impulse train. Third, the reflection coefficients are affected by both vocal tracts. Even if the impulse trains could be properly assigned, there would be the problem of resolving the lattice filter coefficients in some way, so as to reconstruct each speaker's speech separately, without influence from the shape of the competing speaker's vocal tract. Because of these difficulties, further LPC research was not pursued during the project.

SECTION 5
SYLLABLE SORTING

The idea of syllable sorting is to assign each section of mixed voice data to one or the other of two output channels, depending on which speaker's voice is dominant in the data section. No attempt is made to split a sound into parts. Listening tests showed that the method can separate mixed voices fairly well if a good sorting criterion is available. Such criteria were developed for two special cases. The resulting speech processing is discussed in sections 5.1 and 5.2.

5.1. SORT BY PITCH ALGORITHM

One obvious special case which allows assigning sounds to speakers is one where the speakers have distinct pitch ranges. This might typically be found in mixed speech from a male and a female speaker. For each section of data, the predominant pitch may be determined by complex correlation with a search for the peak, as described in sections 3.1 and 3.2. One advantage of using the complex correlation is that it is robust against frequency shift of the signal, such as could result from single sideband (SSB) mistuning.

Sorting by pitch was implemented using overlapping Hanning (raised cosine) time windows applied to the data. The sampling rate was 6400 Hertz and the window overlap was 50 percent. Window lengths of 256 and 512 points (40 and 80 milliseconds) were tried. Each window was assigned to one of two output channels, depending on the window's dominant pitch. The output channel which was rejected received a section of white noise, at about 6 dB below the mixed speech signal. This white noise was

inserted to prevent gaps in the output channels. It is difficult for a human to interpret speech interrupted by silences.

The algorithm separated speech well, provided that the speakers had distinct pitch ranges. Use of the 80 millisecond windows was found to be superior, because occasionally the algorithm would switch back and forth rapidly between the output channels. The algorithm with the 40 millisecond windows would then produce a rapid (50 Hertz) fluttering effect, which was annoying and distracting.

To determine how well the windows were being sorted, an ideal syllable sorter was programmed. This ideal sorter was given two separate voice files, which it then merged at a specified power ratio. That is, th program was told how many dB the first voice should be above the second in the mixed file. The program then computed the mixed voice signal and compared it versus the two separate voice files. For each window of data, the program computed the gain which would minimize the mean square error (mse) between the separate voice file and the mixed voice file. The program then computed mse's and assigned the data accordingly.

The optimal gain to minimize mean square error may be computed as follows. For two real functions f and g define the inner product of f and g as

$$\langle f,g \rangle = \sum_{n=1}^{N} f(n)\, g(n)\, 0.5(1 - \cos(2\pi n/N))$$

Define the norm $||f|| = \sqrt{\langle f,f \rangle}$

It may be verified that these are a true inner product and norm. Then given a solo voice signal f and a mixed voice signal g, we wish to find the gain constant a which minimizes the mse, given by

$$\text{mse} = ||af - g||^2 = \langle af-g,\ af-g\rangle = a^2||f||^2 - 2a\langle f,g\rangle + ||g||^2.$$

Here we make use of the property that $\langle f,g\rangle = \langle g,f\rangle$.

Then

$$\frac{d(\text{mse})}{da} = 2a||f||^2 - 2\langle f,g\rangle$$

and

$$\frac{d^2(\text{mse})}{da^2} = 2||f||^2.$$

The second derivative is non-negative, so the mse has its minimum where its first derivative equals zero. Let a_o be the a which minimizes the mse. Then

$$a_o = \langle f,g\rangle/||f||^2$$

and the minimum mse is

$$\min_a(\text{mse}) = ||g||^2 - \langle f,g\rangle^2/||f||^2.$$

The power of the modified solo voice signal $a_o f$ is $a_o^2||f||^2$, so the best-fit signal to noise ratio (SNR) is

$$\text{SNR} = \frac{\langle f,g\rangle^2/||f||^2}{||g||^2 - \langle f,g\rangle^2/||f||^2}.$$

So

$$\text{SNR} = \frac{\langle f,g\rangle^2}{||f||^2||g||^2 - \langle f,g\rangle^2}$$

provided $||f||^2||g||^2 > 0$.

The ideal syllable sorter was given a tolerance in dB. It sorted each mixed voice window into the output channel representing the speaker with the higher SNR for the window. It assigned the mixed voice signal

also to the alternate output channel if both SNR's were within the tolerance of each other. If the SNR's differed by more than the tolerance, then the alternate output channel was given white noise.

For voices mixed at 0 dB relative levels, it was found that ideal sorting with 10 dB tolerance allowed nearly complete reconstruction of each voice, with only a small amount of the alternate voice coming through on each channel. Thus syllable sorting alone is able to recover all of a person's speech that is no more than 10 dB below a competing talker's speech. This implies that to improve on syllable sorting, a program must pull out speech sounds that are more than 10 dB below the competing sounds from another speaker.

5.2. SORT BY MISTUNING ALGORITHM

A second special case which allows assigning sounds to speakers can occur when the voice signals are transmitted by single sideband (SSB) radio. For SSB reception, it may happen that one voice is received with an amount of mistuning differing from that of the other. The use of complex correlation then gives evidence related to voice frequency shift, and hence to voice mistuning. If two voices are being received with suitably different mistunings, and if the amounts of the two mistunings are known, then the voices can be sorted. Section 6.2 describes a program developed during the present project which automatically estimates pitch and mistuning for single voice speech. The program also produces histograms in graphic form that can help characterize noisy and mixed voice signals.

As shown in Section 3.3, the complex correlation of voiced speech data may be used to determine what raised sinusoid comb best fits the speech power spectrum. For C(n) the complex correlation, and T the value

of n (within the range of possible pitch periods) that maximizes C(n), then the best comb F is

$$F(k) = 0.5 + 0.5 \cos(2\pi Tk + d)$$

where

$$d = \arctan(-\text{Im } C(T)/\text{Re } C(T)) .$$

For properly tuned speech, d should be zero. This is because the peaks in the spectrum are all harmonics of the pitch. The comb which best fits the spectrum will therefore have a maximum (a tooth) at zero frequency. Deviation of the lowest frequency tooth from zero frequency indicates speech mistuning.

A pitch period of T sample intervals is a pitch period of T/F_s seconds, where F_s is the sampling frequency in Hertz. The pitch is therefore F_s/T Hertz. Speech mistuned by H Hertz is therefore mistuned by H/pitch = HT/F_s times the pitch. This is $2\pi HT/F_s$ radians in the cosine term of F(k). The comb itself is displaced by -d radians. Therefore the phase angle between a comb fitted to a mistuning of H Hertz and the actual comb is $(d + 2\pi HT/F_s)$ modulo 2π. When this phase angle is small (or close to a multiple of 2π), the observed data is consistent with the hypothesis that a frequency displacement of H Hertz has taken place. This is the basis of the sort by mistuning algorithm.

The sort by mistuning program is given two mistunings to check for. It then goes through the data, and for each time window determines the best comb filter fit. It checks the phase of each comb to see which mistuning better predicts the phase angle of the comb. For each of the two mistunings there is an output channel. Each data interval is sent to the output channel whose mistuning better predicts the comb phase. If neither mistuning predicts the comb phase well, then the data is sent to both output channels. If an output channel does not receive data, then

the gap is filled with white noise. The sort by mistuning had the usual parameters, namely 6400 Hertz sampling rate, 512 point (80 milliseconds) raised cosine time windows, and time window overlap of 50 percent.

As the preceding discussion implies, mistunings can best be used to separate data if they predict distinctly different comb phase angles, most of the time. This is most likely if both speakers speak with about the same pitch, and the difference in mistunings is about half of the common pitch. Listening tests confirmed that under this condition the sort by mistuning algorithm separates voices well.

The data sorting algorithms separated voices well under their enabling conditions, but there was clearly room for improvement. The white noise introduced was distracting, and occasional absences of data were noticeable. The idea suggested itself to apply the comb filtering described in Section 3. The data sorting criterion would then be used to switch between comb (enhancement) and rake (suppression) filtering, instead of switching between data and noise. Section 6 discusses both this algorithm and a program to automatically estimate mistuning.

SECTION 6
SYLLABLE COMBING AND MISTUNING ESTIMATION

The programs described in Sections 3 and 5 were combined. The result was the best voice processing developed during the project. This voice processing was used to make the project demonstration tape. The voice processing was aided by an automated method to estimate pitch and SSB mistuning.

6.1. SYLLABLE COMBING VOICE PROCESSOR AND RESULTS

Two syllable-combing programs were written. Each operates on data sampled at 6400 Hz. Each uses 512 point (80 millisecond) raised cosine (Hanning) windows overlapped 50 percent. Each produces only one output channel. Each combs (enhances) or rakes (suppresses) each data window depending on a selection criterion. These programs are named FRCOMB and MTCOMB, and are listed in Appendix B. FRCOMB selects by pitch frequency. The operator tells the program the pitch range to check for. Windows with pitches in this range are combed. All other windows are raked. MTCOMB selects by mistuning. The operator tells the program what frequency shift to check for. The operator also tells the program what percent of combing to use. This percent combing indicates what complex correlation phase angles are to trigger data combing. For example, for 50 percent combing the program would check each window for a comb phase angle within 50 percent of π radians (the maximum possible angular distance) of the predicted phase angle. If the phase angle is smaller than the limit, then the window is combed. Otherwise the window is raked.

The program which merges two voice data channels is called WTMERG and is listed in Appendix B. A second voice merging program was written which adds white Gaussian noise at a specified gain level. This program is

called WTNMRG, and is also listed in Appendix B. Listening tests showed that the voice combing algorithms are robust against white Gaussian noise.

The syllable combing programs separated voices well when their selection criteria were valid. Each raked (suppressed) voice was still slightly audible, but sounded faint and whispery. Occasionally the program would suppress portions of the voice which should have been enhanced, but the traces that remained helped the ear fill in the gap and maintain continuity.

When trying to enhance received single sideband data it helps to know the mistuning. Similarly, when trying to sort by pitch it helps to know speaker pitch. A program to determine pitch and mistuning is discussed in the next subsection.

6.2. AUTOMATIC SSB MISTUNING ESTIMATOR

As described in Section 5.2, a given best-fit comb phase is consistent with many possible mistunings. For F_m a possible mistuning frequency and F_p the pitch frequency, then $F_m \pm F_p$ is also a possible mistuning frequency. Thus given one data window there are multiple solutions to the mistuning estimation problem. However, speaker pitch will generally vary over time. For a changed pitch, the new multiple solutions will tend not to line up with the old multiple solutions, except at the true mistuning. This can be used to estimate mistuning.

Program PMCES is a pitch and mistuning estimator, and is listed in Appendix B. The program operates with the usual parameters, namely a 6400 Hertz sampling rate, and 512 point (80 millisecond) raised cosine (Hanning) data windows. The windows are overlapped by 50 percent. A program was also tried with 256 point (40 millisecond) windows. No major difference in program results was observed.

The program compiled a two dimensional histogram, representing estimated mistuning versus estimated pitch. There were 18 pitch bins, representing 80 to 90 Hertz, 90 to 100 Hertz, etc., through 250 to 260 Hertz. There were 161 mistuning bins, representing -402.5 to -397.5, -397.5 to -392.5, etc., through 397.5 to 402.5 Hertz. This gave an 18 x 161 = 2898 bin histogram. For each time window, the program computed the complex correlation. It then found the correlation magnitude peak in the range of 80 to 256 Hertz. From the peak, the program estimated speaker pitch, and also estimated the multiple solutions to the mistuning estimation problem. Then the program incremented the corresponding bins in the pitch-mistuning histogram.

The histogram increment used was the magnitude squared of the complex correlation peak. Using this increment gave relatively less weight to windows whose spectra did not have the comb-like structure typical of voiced speech. This increment also is related to the power in a window, because the complex correlation is proportional to speech amplitude. A program was also written which incremented the histogram by window power. Incrementing by the correlation peak was found to give a cleaner plot when applied to very noisy data.

In addition to incrementing the histogram, the program also calculates the weighted average of the estimated pitch and the weighted average of the square of the estimated pitch. The weighting used is the same as the histogram increment. When all the data is processed, the program computes and prints the mean and standard deviation of estimated speaker pitch. It also estimates mistuning by finding the peak of a one-dimensional mistuning histogram. Finally, it plots the pitch-mistuning histogram.

Figures 6-1 and 6-2 show some plots that this program produced. Figure 6-1 results from processing 37 seconds of microphone speech of a male speaker. Mistuning is plotted horizontally and pitch is plotted

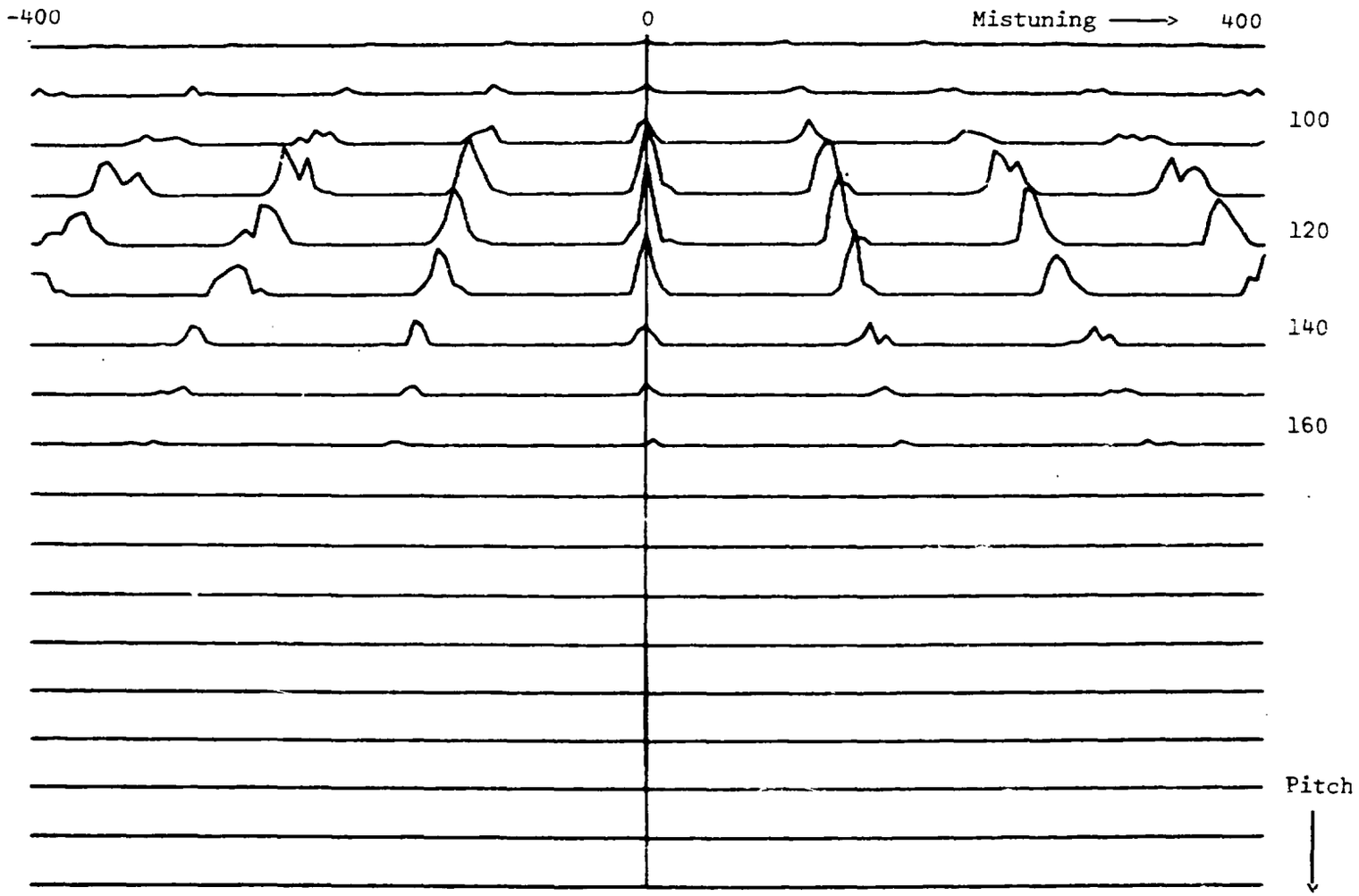


Figure 6-1 Microphone Speech Pitch Versus Mistuning

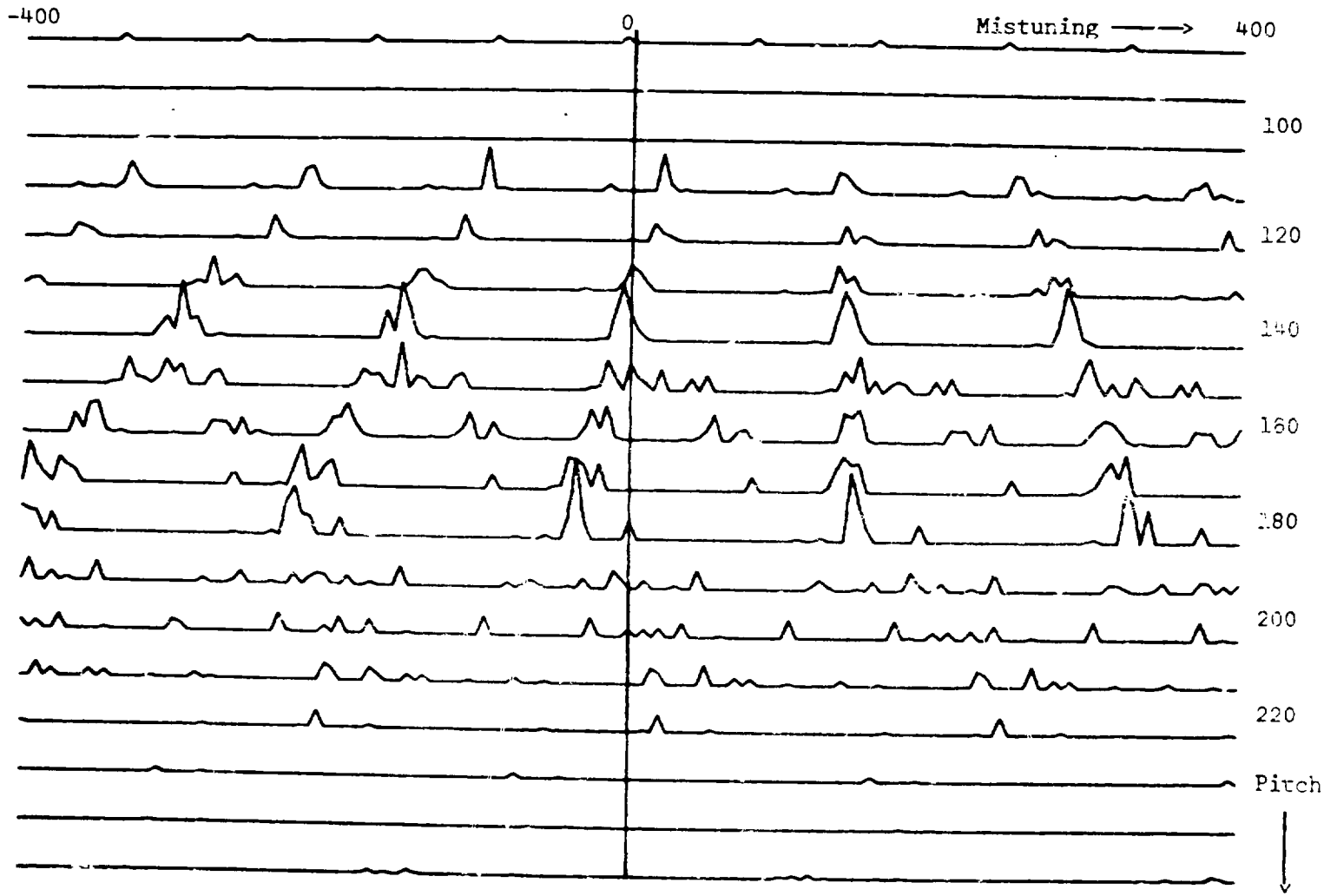


Figure 6-2 SSB Speech Pitch Versus Mistuning

vertically. The pitch is seen to vary mainly between 100 to 150 Hertz. On each line, multiple solutions for the mistuning are visible. However, the only mistuning indication common to all lines is at the vertical axis, which represents zero Hertz mistuning. It is typical of these pitch-mistuning plots that the peaks form slanting lines for false mistuning solutions and a vertical line for the true solution. Sometimes this property can be used to determine the true mistuning by eye, for very noisy data that fools the automatic mistuning calculation.

Figure 6-2 results from reception of amateur single sideband (SSB) under moderately noisy conditions. It is for ten seconds of data during which the speaker said "... uniform, W2PAU, five nine zero five, five nine zero five." This plot is not as clean as Figure 6-1, but the speaker's pitch and mistuning are clearly visible. The speaker's pitch was 110 to 190 Hertz. The frequency shift of the received signal was 145 Hertz, indicated by the vertical line of peaks.

These results show that sorting by pitch or mistuning does not have to be based on guesswork. An automated method is available for estimating pitch and mistuning.

SECTION 7
CONCLUSIONS AND RECOMMENDATIONS

As Sections 2 through 6 have shown, the main positive results of the present effort were based on the use of complex correlation. Adaptive lattice filtering did not lead to useful results. The complex correlation allowed automatic adaptive design of comb filters to enhance and suppress data. The complex correlation also gave estimates of speaker pitch, and could be used to estimate speaker frequency shift. It also turned out to be robust against noise and distortion.

The voice separation algorithms developed during the project apply to special cases. The main obstacle to treating the general case was the lack of a general method to identify to which speaker the various sounds in mixed speech belong. Identifying by adaptive lattice filtering coefficients (Section 2) was found to be impractical. Identifying by power level (Sections 3 and 4) was also found to be unsatisfactory. Voice separation was achieved for the special cases of distinct speaker pitches and distinct frequency shifts (Sections 5 and 6).

A program for automatically estimating speech frequency shift was developed in this project. This may be a new result. In itself, it may aid in improving intelligibility, or at least it may be used to improve single sideband reception quality. The estimator can be used to estimate the mistuning of baseband signals, and thereby direct their retuning.

Whether or not the techniques developed will aid in any particular radio reception problem may depend on the details and specifics of the problem. There are thus two ways to go in the future. One way is to take the algorithms to the problems. By this is meant developing software (and possibly host hardware) which is convenient for data analysts to use

and apply to their particular problems. The algorithm developed to separate speech can be programmed to run in real time, if a dedicated FFT or array processor is used.

A second way to proceed in the future is to bring the problems to the algorithms. By this is meant supplying some target data for processing with the existing software, perhaps with modifications to better fit the data. Each approach has its own advantages. Experienced data analysts are more likely to appreciate the results of signal processing. On the other hand, the second approach is much less costly, and would allow signal processing by personnel familiar with the algorithms developed.

REFERENCES

1. John Makhoul, "Stable and Efficient Methods for Linear Prediction," IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. ASSP-25, pp. 423-428, October 1977.

2. John D. Markel, Beatrice T. Oshika, and Augustine H. Gray, Jr., "Long-Term Feature Averaging for Speaker Recognition," IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. ASSP-25, pp. 330-337, August 1977.

3. Alan V. Oppenheim and Ronald W. Schafer, Digital Signal Processing, Prentice-Hall, Englewood Cliffs, NJ, 1975.

4. Lawrence R. Rabiner and Bernard Gold, Theory and Application of Digital Signal Processing, Prentice-Hall, Englewood Cliffs, NJ, 1975.

APPENDIX A

SSB MISTUNING SIMULATOR

A program was written to perform digitally the frequency shifting needed to simulate single sideband (SSB) mistuning. The program was also useful for correcting the mistunings of actual SSB radio data. The program is named WMTSSB, and is listed in Appendix B.

The heart of the program is a 90 degree phase shifter, also known as a Hilbert transformer. This is a finite impulse response (FIR) digital filter. It was designed using an FIR filter design program (using the Remez exchange algorithm) which was taken from Rabiner and Gold [4]. A real waveform is always made up of positive and negative frequency components. However, an imaginary part can be attached to a pure real waveform, in order to make the waveform have only positive frequency components. This process simulates in baseband the sideband suppression filtering normally done at IF in an SSB transmitter.

Lower sideband suppression may be done by applying a linear filter with frequency response $G(e^{j\omega}) = 0.5 + 0.5jH(e^{j\omega})$ where

$$H(e^{j\omega}) = \begin{cases} -j, & 0 \leq \omega < \pi \\ j, & \pi \leq \omega < 2\pi \end{cases}$$

Here we are working with a sampled-time system, so $H(e^{j\omega})$ is the z transform of the impulse response h(n) with $z = e^{j\omega}$. Then as Rabiner and Gold show [4, p. 71]

$$h(n) = \begin{cases} \dfrac{2 \sin^2(\pi n/2)}{\pi n}, & n \neq 0 \\ 0, & n = 0 \end{cases}$$

This is the ideal digital Hilbert transform filter. Rabiner and Gold include in their book (pp. 194 to 204) a program that can design finite impulse response (FIR) approximations to the ideal Hilbert transformer. This filter design program was implemented by PAR on a previous project.

The filter design program was used to design an equiripple approximation to $H(e^{j\omega})$, for the passband of frequencies from 0.03 F_s to 0.47 F_s, where F_s is the sampling frequency (6400 Hertz for the present project). This is a passband of 192 to 3008 Hz. A 31 tap filter was designed which turned out to be within $\pm$ 0.21 dB of the ideal within the passband. All the even numbered taps were zero. The odd numbered taps are given in Table A-1. The taps of the table are related to the ideal h(n) by g(n+16) approximates -h(n).

Table A-1 Hilbert Transformer Tap Weights

Tap		Weight		Tap
g(1)	=	0.02050	=	-g(31)
g(3)	=	0.02134	=	-g(29)
g(5)	=	0.03265	=	-g(27)
g(7)	=	0.04876	=	-g(25)
g(9)	=	0.07296	=	-g(23)
g(11)	=	0.11398	=	-g(21)
g(13)	=	0.20402	=	-g(19)
g(15)	=	0.63385	=	-g(17)

The frequency shifting program used the Hilbert transformer to zero the negative frequency components of the signal. Next it doubled the sampling rate for the (now complex) signal. This was done so that

frequency shifting would not result in aliasing. For example, a sinusoid at 3 kHz shifted up 400 Hertz to 3,4 kHz would appear not to be shifted at all, if the sampling rate were only 6.4 kHz. This is the well known "aliasing" effect for sampled-time signals. The sampling rate was doubled by first inserting a zero between each pair of consecutive samples. Then the signal was passed through a six pole Butterworth lowpass filter, which smoothed the data. The filter cutoff was 0.2 of the new (doubled) sampling rate. This is 2560 Hertz. Next the signal was multiplied by a complex exponential, producing a frequency shifted signal. Only the real part of the result was saved. Next the signal was lowpassed with the same Butterworth filter used previously. This prevented aliasing. Finally, the sampling rate was halved, that is, returned to its original value.

To summarize, the steps in the frequency shifter were

1. Attach j times the Hilbert transform to the signal.
2. Double the sampling rate by interpolating zeroes.
3. Smooth the data by lowpass filtering.
4. Multiply by a complex exponential to shift frequency.
5. Discard imaginary part.
6. Lowpass again to prevent aliasing.
7. Discard alternate samples.

The Hilbert transformer was implemented so that tap 16 represented time zero of the impulse response. This made it a "non-realizable" filter, which predicts the future. This was done to avoid introducing delay in the signal when it was aligned with the imaginary signal that was attached to it.

Figure A-1 shows in graphic form the result of frequency shifting by +200 Hertz. Lines in the figure are organized in groups of three. Each line consists of four frames, with each frame of each line normalized.

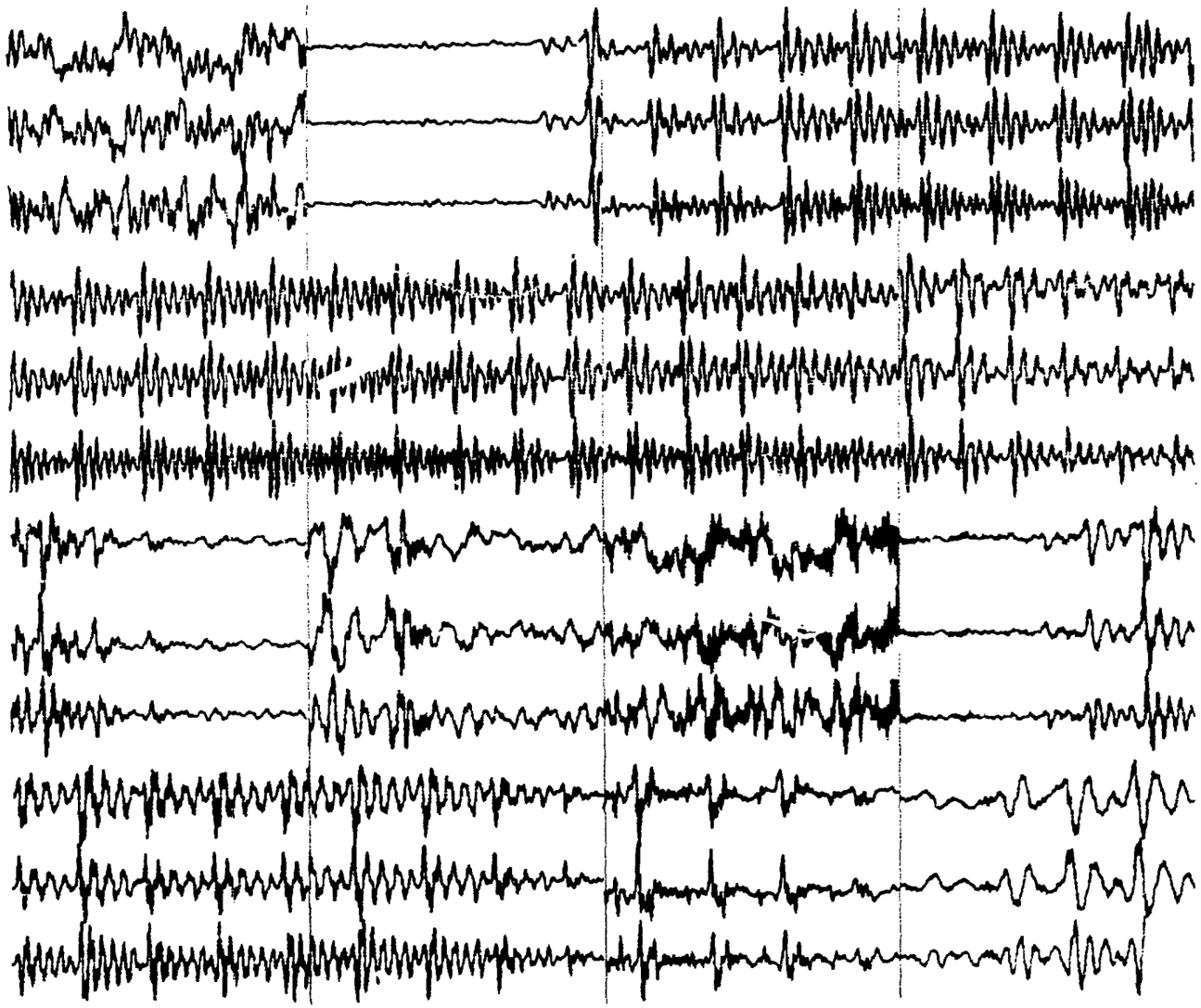


Figure A-1 Frequency Shifted Speech

Each frame has 256 points (40 milliseconds) of data. The first line of each group shows the original data. The second line of each group shows the imaginary part attached to the data. Note that this is in phase quadrature with the original data. That is, the second line of each group has a zero crossing where the first line has a peak, and vice versa. The third line shows the result of shifting the data up 200 Hertz. Plots such as this helped verify the correct operation of the program.

APPENDIX B
COMPUTER SOFTWARE

The software to be presented was developed to run on a Digital Equipment Corporation PDP-11/45 computer with a Tektronix 4014-1 alphanumerics/graphics terminal and a Tektronix 4631 hard copy unit. The software runs under the RSX-11M operating system, version 3.1. All the software developed during the project was compiled under Fortran IV Plus. The software is believed to be compatible with Fortran IV, but this has not been checked. Several subroutines were used for input and output that were not developed during the project, and are not listed. These are READF, WRITEF, PLOTS, PLOT, and LINE.

READF and WRITEF are fast programs to read and write data to and from disk. Their calling formats are

```
CALL WRITEF(LUN, BLKNO, BUF, SIZE, IOSB)
CALL READF (LUN, BLKNO, BUF, SIZE, IOSB)
```

where
- LUN is the logical unit number (integer *2)
- BLKNO is the virtual block number (integer *4)
- BUF is the data buffer (vector of integer *2)
- SIZE is the data block size in bytes (integer *2)
- IOSB is the I/O status block (2 element integer *2 vector)

PLOTS is called with no arguments. It erases the Tektronix 4014 screen, puts it in graphics mode, and sets the plotting origin at the lower left corner. PLOT is used in the project software to move the plotting origin. The calling format is

```
CALL PLOT (X, Y, -3)
```

where X is the horizontal shift of the origin (type real)
Y is the vertical shift of the origin (type real)

LINE is used to plot a line of data. That is, LINE plots a series of points and connects them with straight lines. The calling format is

CALL LINE (XBUF, YBUF, NPTS, 1, 0, 0)

where XBUF is the vector of X coordinates (vector of reals)
YBUF is the vector of Y coordinates (vector of reals)
NPTS is the number of points to plot (integer *2)

XBUF and YBUF must each have NPTS+2 components. XBUF (NPTS+1) and YBUF (NPTS+1) each contains the starting value of the line (usually the minimum value in the line). XBUF (NPTS+2) and YBUF(NPTS+2) tell the number of coordinate-units per distance-unit.

The format of the project waveform files was as follows. Each file consisted of a series of 512-byte blocks. The first block of each file was a header block. This was followed by a variable number of data blocks, each containing 256 integer *2 words. The header block format in integer *2 words was as follows.

Word(s)	Value	Meaning
1	1	One channel of data
(2,3)	(6400,0)	Sampling rate 6400 Hertz
(4,5,6)	-	Start time in hours, minutes, seconds
(7,8,9)	-	Stop time in hours, minutes, seconds
(10-13)	-	Not used
14	0	Data is integer *2
(15,16)	-	Not used
(17,18)	(NLO,NHI)	There are 65536*NHI+NLO data blocks.

Not all programs maintained the start and stop times. Words 17 and 18 sometimes appear in the programs as word 9 of an integer *4 array.

The software produced during the program was written using structured programming, with statement indenting to show program structure. Use of this method was found to greatly aid program readability and reliability.

The program listings have hand-drawn flow-of-control arrows added. This makes each listing, in effect, its own flowchart. Programs are listed in the alphabetical order of the main programs in which they appear. The exceptions are files CFFT and HILB. File CFFT contains the complex fast Fourier transform (CFFT) subroutine. File HILB contains two subroutines used by the WMTSSB program. These are HILB and LWPS, the Hilbert transformer (90 degree phase shifter) and Butterworth lowpass filter respectively.

Table B-1 gives a table of contents for the listings. SBR indicates that a program is a subroutine.

Table B-1 Guide to Software Listings

Program	Function
CCORPL	Complex correlation and plot
CFFT	Complex fast Fourier Transform (FFT) SBR
FRCOMB	Comb by pitch frequency
HILB	Hilbert transformer SBR, lowpass SBR
MTCOMB	Comb by (SSB) mistuning
PMCES	Pitch-mistuning estimator
VCRFSP	Voice lattice filter, threshold, reconstruct
WMTSSB	"Write mistuned SSB" (Frequency shifter)
WTMERG	Does weighted merge of waveform files
WTNMRG	Does weighted merge, adds white Gaussian noise

```
C-----CCORPL BY BOB DICK
C-----FROM TMSPPL 14 AUG 80
C-----COMPLEX CORRELATION MAGNITUDE PLOT
C-----ONE SECOND OF DATA PER PAGE.
      BYTE FNAM(30),ANSW
      DIMENSION DATIN(256),WNDW(256),TR(256),TI(256)
      DIMENSION AMPL(52),XC(52),IBUF(256)
      INTEGER *4 IREC,IOSB,IBF(9)
      EQUIVALENCE (IBF,IBUF)
      DATA DATIN/256*0./PI/3.14159265/
      TYPE 10010
      ACCEPT 10020,LNTH,(FNAM(KK),KK=1,LNTH)
      FNAM(LNTH+1)=0
     OOPEN(UNIT=1,NAME=FNAM,TYPE='OLD',
     1     BUFFERCOUNT=-1,ERR=9000,READONLY)
      IREC=1
      CALL READF(1,IREC,IBUF,512,IOSB)
      N1BLKS=IBF(9)
      TYPE 10030,N1BLKS
      ACCEPT 10040,NFBLK
      IF(NFBLK.LE.0)GO TO 9000
      ACCEPT 10040,NMBLKS
      NLBLK=NFBLK+NMBLKS-1
      IF(NLBLK.LT.NFBLK)GO TO 9000
      IF(NLBLK.GT.N1BLKS)GO TO 9000
C-----SETUP HANNING DATA WINDOW
      DO 10 IDX=1,256
      WNDW(IDX)=0.5-0.5*COS(PI*IDX/128.)
   10 CONTINUE
C-----SET X COORDS FOR AMPL LINE
      DO 20 IDX=1,50
      XC(IDX)=51-IDX
   20 CONTINUE
      XC(51)=1
      XC(52)=1./0.23
C-----SET Y COORDS FOR CORR LINES
      CALL STYC
      NFIRST=NFBLK*2-1
C-----SKIP FIRST HALF-WINDOW
      CALL DA128(DATIN,NFIRST)
      NMIN=NFIRST+1
      LAST=NLBLK*2
C-----DO UNTIL (NO MORE DATA) PLOT PAGES
   30 CONTINUE
C--------DONE ONCE PER PAGE
      CALL PLOTS
      DO 40 IDX=1,50
      AMPL(IDX)=0.
   40 CONTINUE
      DO 90 IXLN=1,50
C-----------DONE ONCE PER CORRELATION LINE
C-----------GET NEW DATA
      CALL DA128(DATIN,NMIN)
      PW=0.
      DO 50 IDX=1,256
      TR(IDX)=DATIN(IDX)*WNDW(IDX)
      PW=PW+TR(IDX)*DATIN(IDX)
      TI(IDX)=0.
```

ABORT

ABORT

ABORT

```
   50   CONTINUE
        AMPL(IXLN)=SQRT(PW)
        CALL CFFT(1,TR,TI,8)
C-------FORM AMPLITUDE SPECTRUM
        DO 60 IDX=1,128
          TR(IDX)=SQRT(TR(IDX)*TR(IDX)+TI(IDX)*TI(IDX))
          TI(IDX)=0.
   60   CONTINUE
C-------ZERO NEGATIVE FREQUENCY PORTION
        DO 70 IDX=129,256
          TR(IDX)=0.
          TI(IDX)=0.
   70   CONTINUE
C-------TAKE INVERSE FFT
        CALL CFFT(-1,TR,TI,8)
C-------FORM MAGNITUDE OF COMPLEX CORRELATION
        DO 80 IDX=1,128
          TR(IDX)=SQRT(TR(IDX)*TR(IDX)+TI(IDX)*TI(IDX))
   80   CONTINUE
C-------PLOT A CORRELATION LINE
        CALL CCRPLT(TR,IXLN)
        NMIN=NMIN+1
C-------IF(NO MORE DATA) EXIT CORR LINE LOOP
        IF(NMIN.GT.LAST)GO TO 100
    90  CONTINUE
   100  CONTINUE
C-------PLOT AMPLITUDE LINE
        CALL FCTR(AMPL,52)
        AMPL(52)=AMPL(52)/0.8
        YORG=9.9
        CALL PLOT(0.,YORG,-3)
        CALL LINE(XC,AMPL,50,1,0,0)
C-------REPEAT LINE TO FILL BUFFER
        CALL LINE(XC,AMPL,50,1,0,0)
        YORG=-YORG
        CALL PLOT(0.,YORG,-3)
C-------WAIT FOR SIGNAL
        ACCEPT 10050,ANSW
C-----IF(X TYPED) EXIT PAGE LOOP
      IF(ANSW.EQ.'X')GO TO 110
C-----IF(MORE DATA)REPEAT PAGE LOOP
      IF(NMIN.LE.LAST)GO TO 30
  110 CONTINUE
 9000 CONTINUE<-------------------ABORT
      STOP
10010 FORMAT(' COMPLEX CORRELATION PLOTTER INPUT FILE?')
10020 FORMAT(Q,30A1)
10030 FORMAT(1X,I8,' BLOCKS.  GIVE FIRST <CR> HOW MANY')
10040 FORMAT(I8)
10050 FORMAT(A1)
      END
-------------------------------------------------------------
      SUBROUTINE RA128(DAT,KALL)
C-----BY BOB DICK 8 AUG 80
C-----ADVANCES DATA 128 PTS/CALL
      DIMENSION DAT(256),IBUF(256)
      INTEGER*4 IREC,IOSB
      IF(.NOT.(KALL.EQ.2*(KALL/2)))GO TO 20
C-------HERE CALL IS EVEN, USE CURRENT BUFFER
```

```
      DO 10 IDX=1,128
      DAT(IDX)=DAT(IDX+128)
      DAT(IDX+128)=IBUF(IDX+128)
   10 CONTINUE
      GO TO 40
   20 CONTINUE
C-----HERE CALL IS ODD, USE NEW BUFFER
      IREC=KALL/2+2
      CALL READF(1,IREC,IBUF,512,IOSB)
      DO 30 IDX=1,128
      DAT(IDX)=DAT(IDX+129)
      DAT(IDX+128)=IBUF(IDX)
   30 CONTINUE
   40 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE CCRPLT(DATA,LIN)
C-----BY BOB DICK
C-----FROM SPCPLT 14 AUG 80
C-----PLOTS A LINE OF COMPLEX CORRELATION
      DIMENSION DATA(130),YC(130)
      CALL FCTR(DATA,130)
      DATA(130)=DATA(130)/0.69
      XORG=(50-LIN)*0.23
C-----0.23*49+0.69=11.96 INCHES HORIZ
      CALL PLOT(XORG,0.,-3)
      CALL LINE(DATA,YC,128,1,0,0)
      XORG=-XORG
      CALL PLOT(XORG,0.,-3)
      RETURN
      ENTRY STYC
      DO 1010 IDX=1,128
      YC(IDX)=128-IDX
 1010 CONTINUE
      YC(129)=0.
C-----128 POINTS AT 13/INCH IS 9.8 INCHES
      YC(130)=13.
      RETURN
      END
```

```
      SUBROUTINE FCTR(FNC,IDIM)
      DIMENSION FNC(258)
      IF(.NOT.(IDIM.GE.4))GO TO 900 ----------> ABORT
      IF(.NOT.(IDIM.LE.258))GO TO 900 ----------> ABORT
      FMN=FNC(1)
      FMX=FNC(1)
      DO 10 IDX=2,IDIM-2
      IF(FMN.GT.FNC(IDX))FMN=FNC(IDX)
      IF(FMX.LT.FNC(IDX))FMX=FNC(IDX)
   10 CONTINUE
      FNC(IDIM-1)=FMN
      FNC(IDIM)=FMX-FMN
  900 CONTINUE <---------- ABORT
      RETURN
      END
```

```
      SUBROUTINE CFFT(MODE,XR,XI,M)
C-----BY BOB DICK.  ENTERED 14 FEB 80.
C-----FAST FOURIER TRANSFORM WITH COMPLEX INPUT
C-----MODE+=FORWARD, MODE-=BACKWARD
C-----XR IS REAL VECTOR, XI IS IMAG, DIMENSION IS 2**M.
C-----DECIMATION IN TIME ALGORITHM, OPPENHEIM AND SCHAFER
C-----FIG 6.10 P 297.  BUGGY PROGRAM ON P 331.
      DIMENSION XR(1024),XI(1024)
      DATA PI/3.14159265358979/
      N=2**M
      NV2=N/2
C-----ORDER DATA BY BIT-REVERSED INDEX
      NM1=N-1
      IBTRV=1
      DO 40 IDX=1,NM1
        IF(.NOT.(IDX.LT.IBTRV))GO TO 10
          TMPR=XR(IBTRV)
          TMPI=XI(IBTRV)
          XR(IBTRV)=XR(IDX)
          XI(IBTRV)=XI(IDX)
          XR(IDX)=TMPR
          XI(IDX)=TMPI
   10   CONTINUE
C-------INCREMENT THE BIT-REVERSED INDEX
        IPTR=NV2
        GO TO 30
   20   CONTINUE
          IBTRV=IBTRV-IPTR
          IPTR=IPTR/2
   30   CONTINUE
        IF(IBTRV.GT.IPTR)GO TO 20
        IBTRV=IBTRV+IPTR
   40 CONTINUE
C-----PERFORM THE BUTTERFLIES
      LEAP=1
      DO 70 ISTG=1,M
        JUMP=LEAP
        LEAP=LEAP*2
        COUNT=JUMP
        TWSTR=COS(PI/COUNT)
        TWSTI=SIN(PI/COUNT)
        IF(MODE.GE.0)TWSTI=-TWSTI
        TURNR=1.
        TURNI=0.
        DO 60 IDX=1,JUMP
          DO 50 IPTR=IDX,N,LEAP
            JPTR=IPTR+JUMP
            TMPR=XR(JPTR)*TURNR-XI(JPTR)*TURNI
            TMPI=XR(JPTR)*TURNI+XI(JPTR)*TURNR
            XR(JPTR)=XR(IPTR)-TMPR
            XI(JPTR)=XI(IPTR)-TMPI
            XR(IPTR)=XR(IPTR)+TMPR
            XI(IPTR)=XI(IPTR)+TMPI
   50     CONTINUE
          TMPR=TURNR*TWSTR-TURNI*TWSTI
          TURNI=TURNR*TWSTI+TURNI*TWSTR
          TURNR=TMPR
   60   CONTINUE
```

```
   70 CONTINUE
      IF(.NOT.(MODE.LT.0))GO TO 90
         FCTR=1./N
         DO 80 IDX=1,N
            XR(IDX)=XR(IDX)*FCTR
            XI(IDX)=XI(IDX)*FCTR
   80    CONTINUE
   90 CONTINUE
      RETURN
      END
```

```
C-----FRCOMB BY BOB DICK
C-----LAST UPDATE 10 SEP 80
C-----FROM MTCOMB 4 SEP 80
C-----COMB FILTERS VOICE DEPENDING ON PITCH FREQUENCY
C-----DATA ARRAYS:
C-----CCR,CCI:  LIN MAG SPECTRUM, THEN COMPLEX CORRELATION
C-----DATIN:  INPUT DATA BEFORE WINDOWING
C-----IBUF:  INPUT BUFFER
C-----KBUF:  HEADER, THEN OUTPUT BUFFER
C-----OBUF:  DATA OVERLAP OUTPUT BUFFER
C-----TR,TI:  WINDOWED DATA, THEN FFT, THEN FILTERED DATA
C-----WNDW:  RAISED COSINE (HANNING) WINDOW
      BYTE F1NAM(30),F2NAM(30)
      DIMENSION DATIN(512),WNDW(512),TR(512),TI(512)
      DIMENSION CCR(512),CCI(512),OBUF(256)
      DIMENSION IBUF(256),KBUF(256)
      INTEGER*4 IREC,IOSB,IBF(9),KBF(9)
      EQUIVALENCE (IBF,IBUF),(KBF,KBUF)
      DATA KBUF/256*0/DATIN/512*0./
      DATA OBUF/256*0./RAKE,COMB/-0.5,0.5/
      DATA PI,TWPI/3.1415926,6.28318531/
      DATA SAMPFQ/6400./
C-----GET INPUT FILE, NUMBER OF BLOCKS
      TYPE 10010
      ACCEPT 10020,LNTH,(F1NAM(KH),KH=1,LNTH)
      F1NAM(LNTH+1)=0
     OOPEN(UNIT=1,NAME=F1NAM,TYPE='OLD',
     1     BUFFERCOUNT=-1,ERR=9000,READONLY)
      IREC=1
      CALL READF(1,IREC,IBUF,512,IOSB)
      N1BLKS=IBF(9)
      TYPE 10030,N1BLKS
      ACCEPT 10040,NMBLKS
      IF(NMBLKS.LT.2)GO TO 9000 ----------> ABORT
      IF(NMBLKS.GT.N1BLKS)GO TO 9000 -----> ABORT
C-----GET OUTPUT FILE, WRITE HEADER
      TYPE 10050
      ACCEPT 10020,LNTH,(F2NAM(KH),KH=1,LNTH)
      F2NAM(LNTH+1)=0
     OOPEN(UNIT=2,NAME=F2NAM,TYPE='NEW',
     1     BUFFERCOUNT=-1,ERR=9000)
      KBUF(1)=1
      KBUF(2)=6400
      SEC=NMBLKS/25.
      MIN=NMBLKS/1500
      ISEC=SEC-60.*MIN
      KBUF(8)=MIN
      KBUF(9)=ISEC
      KBF(9)=NMBLKS
      CALL WRITEF(2,IREC,KBUF,512,IOSB)
C-----SET UP DATA WINDOW
      DO 10 IDX=1,512
      WNDW(IDX)=0.5-0.5*COS(PI*IDX/256.)
   10 CONTINUE
C-----GET PITCH RANGE TO COMB FOR
      TYPE 10060
      ACCEPT 10070,PTLOW
      ACCEPT 10070,PTHIGH
```

```
C-----COMPUTE PITCH PERIOD RANGE FROM PITCH RANGE
      IF(.NOT.(PTLOW.LT.PTHIGH))GO TO 9000 ----------> ABORT
      PERHI=256.
      IF(PTLOW.GT.0.)PERHI=6400./PTLOW
      IF(.NOT.(PTHIGH.GT.0.))GO TO 9000 ----------> ABORT
      PERLO=6400./PTHIGH
C-----ZERO OUTPUT BUFFER
      DO 15 IDX=1,256
        KBUF(IDX)=0
   15 CONTINUE
C-----READ FIRST DATA RECORD BEFORE MAIN LOOP.
      NMIN=1
      CALL TFSADV(DATIN,NMIN)
C-----LOOP NUMBER_OF_RECORDS-1 TIMES.
      DO 80 NMIN=2,NMBLKS
C-----|--MAIN LOOP.  DONE ONCE PER DATA RECORD.
C-----|--TAKE TIME WINDOWS WITH 50 PERCENT OVERLAP.
      |   CALL TFSADV(DATIN,NMIN)
      |   DO 20 IDX=1,512
      |     TR(IDX)=DATIN(IDX)*WNDW(IDX)
      |     TI(IDX)=0.
   20 |   CONTINUE
C-----|--FORM AMPLITUDE SPECTRUM
      |   CALL CFFT(1,TR,TI,9)
      |   DO 30 IDX=1,256
      |     CCR(IDX)=SQRT(TR(IDX)*TR(IDX)+TI(IDX)*TI(IDX))
      |     CCI(IDX)=0.
   30 |   CONTINUE
C-----|--SET NEGATIVE FREQUENCY PORTION TO ZERO
      |   DO 40 IDX=257,512
      |     CCR(IDX)=0.
      |     CCI(IDX)=0.
   40 |   CONTINUE
C-----|--TAKE INVERSE FFT
      |   CALL CFFT(-1,CCR,CCI,9)
C-----|--SEARCH FOR PEAK FROM LOW TO HIGH TIME LIMIT
      |   PEAK=0.
      |   IDPK=25
      |   DO 50 IDX=25,80
      |     TEMP=CCR(IDX)*CCR(IDX)+CCI(IDX)*CCI(IDX)
      |     IF(.NOT.(PEAK.LT.TEMP))GO TO 45
      |       PEAK=TEMP
      |       IDPK=IDX
   45 |     CONTINUE
   50 |   CONTINUE
C-----|--FIND QUADRATIC INTERPOLATION PEAK.
      |   BSQ=CCR(IDPK-1)*CCR(IDPK-1)+CCI(IDPK-1)*CCI(IDPK-1)
      |   USQ=CCR(IDPK+1)*CCR(IDPK+1)+CCI(IDPK+1)*CCI(IDPK+1)
      |   X=1.
      |   IF(PEAK.GT.USQ)X=0.
      |   IF(BSQ.GE.PEAK)X=-1.
      |   IF(X.EQ.0.)X=0.5*(USQ-BSQ)/(2.*PEAK-BSQ-USQ)
      |   TAU=IDPK-1+X
C-----|--INTERPOLATE REAL AND IMAG PARTS (VIA QUADRATIC)
      |   CF1=0.5*(CCR(IDPK+1)-CCR(IDPK-1))
      |   CF2=0.5*(CCR(IDPK-1)+CCR(IDPK+1))-CCR(IDPK)
      |   PARTR=CCR(IDPK)+X*(CF1+X*CF2)
      |   CF1=0.5*(CCI(IDPK+1)-CCI(IDPK-1))
```

```
      CF2=0.5*(CCI(IDPK-1)+CCI(IDPK+1))-CCI(IDPK)
      PARTI=CCI(IDPK)+X*(CF1+X*CF2)
      PARTM=SQRT(PARTR*PARTR+PARTI*PARTI)
      IF(PARTM.LE.0.)PARTM=1.
C-----COMB OR RAKE ACCORDING TO PITCH
      SWITCH=COMB
      IF(TAU.LT.PERLO)SWITCH=RAKE
      IF(TAU.GT.PERHI)SWITCH=RAKE
      OMEG=PI*TAU/256.
      DO 70 IDX=1,256
        CS=(PARTR*COS(OMEG*IDX)+PARTI*SIN(OMEG*IDX))/PARTM
        FILT=0.5+SWITCH*CS
        JDX=513-IDX
        TR(IDX)=FILT*TR(IDX)
        TI(IDX)=FILT*TI(IDX)
        TR(JDX)=FILT*TR(JDX)
        TI(JDX)=FILT*TI(JDX)
   70 CONTINUE
C-----TAKE INVERSE FFT
      CALL CFFT(-1,TR,TI,9)
C-----OVERLAP RESULTING TIME FUNCTIONS
      NMOUT=NMIN-1
      CALL LAPOUT(TR,OBUF,2,NMOUT)
   80 CONTINUE
C-----WRITE FINAL DATA RECORD AFTER MAIN LOOP.
      DO 90 IDX=1,256
        TR(IDX)=0.
   90 CONTINUE
      CALL LAPOUT(TR,OBUF,2,NMBLKS)
 9000 CONTINUE <-------------------- ABORT
  <-- STOP
10010 FORMAT(' PITCH FREQUENCY COMBER INPUT FILE?')
10020 FORMAT(Q,30A1)
10030 FORMAT(1X,I8,' BLOCKS.  HOW MANY DO YOU WANT?')
10040 FORMAT(I8)
10050 FORMAT(' WHAT OUTPUT FILE?')
10060 FORMAT(' WHAT PITCH RANGE?(USE . AND 2 LINES)')
10070 FORMAT(F10.2)
      END
```

```
      SUBROUTINE TFSADV(X,ICALL)
C-----BY BOB DICK.  REVISION OF 14 MAR 80.
C-----ADVANCES BUFFER 256 POINTS PER CALL.
  --> DIMENSION X(512),IBUF(256)
      INTEGER*4 IREC,IOSB
      IREC=ICALL+1
      CALL READF(1,IREC,IBUF,512,IOSB)
      DO 10 IDX=1,256
        X(IDX)=X(IDX+256)
        X(IDX+256)=IBUF(IDX)
   10 CONTINUE
  <-- RETURN
      END
```

```
      SUBROUTINE LAPOUT(DAT,BUF,LUN,KALL)
C-----OUTPUTS DATA WITH 50 PERCENT OVERLAP.
C-----INITIALLY BUF SHOULD BE ALL ZERO.
C-----DOES NOT INCREMENT KALL.
      DIMENSION DAT(512),BUF(256),KBUF(256)
      INTEGER*4 KREC,IOSB
```

```
      DO 10 IDX=1,256
         KBUF(IDX)=DAT(IDX)+BUF(IDX)
         BUF(IDX)=DAT(IDX+256)
   10 CONTINUE
      KREC=KALL+1
      CALL WRITEF(LUN,KREC,KBUF,512,IOSB)
      RETURN
      END
```

```
      SUBROUTINE MILB(DAT,PRES,RES,FRES)
      DIMENSION DAT(256),PRES(15),RES(256),FRES(15)
C-----BY BOB DICK.  15 JAN 80.  UPDATED 16 JAN 80.
C-----RES IS RESULT.  ADJUST ENDS AS FOLLOWS.
C-----ADD PRES TO END 15 POINTS OF PREVIOUS RESULT.
C-----ADD FRES TO FIRST 15 POINTS OF NEXT FUTURE RESULT.
C-----31 POINT FIR FILTER, EVEN TAPS 0, 8 DISTINCT MAGNITUDES.
C-----GAIN 1+OR-0.024 IN 0.03 TO 0.47 SAMPLING RATE.
      DIMENSION VAR(316),OUT(286),TAP(8)
      DATA TAP/.0205,.0213,.0326,.0488,
     2        .0730,.1140,.2040,.6338/
      DO 10 IDX=1,30
      VAR(IDX)=0.
      VAR(286+IDX)=0.
   10 CONTINUE
      DO 20 IDX=1,256
      VAR(IDX+30)=DAT(IDX)
   20 CONTINUE
      DO 40 MID=16,301
      SUM=0.
      DO 30 IPT=1,8
      SUM=SUM+(VAR(MID+2*IPT-1)-
     2         VAR(MID-2*IPT+1))*TAP(9-IPT)
   30 CONTINUE
      OUT(MID-15)=SUM
   40 CONTINUE
      DO 50 IDX=1,15
      PRES(IDX)=OUT(IDX)
      FRES(IDX)=OUT(271+IDX)
   50 CONTINUE
      DO 60 IDX=1,256
      RES(IDX)=OUT(15+IDX)
   60 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE LWPS(FM,DAT)
      DIMENSION FM(6),DAT(512)
C-----LOWPASS 0.2 SAMPLING RATE, 6 POLE PUTTERWORTH.
C-----FM IS FILTER MEMORY.  SHOULD BE 0 INITIALLY.
      DATA AK1,A11,A21/.27725,.49595,-.60494/
      DATA AK2,A12,A22/.20657,.36953,-.19582/
      DATA AK3,A13,A23/.18007,.32212,-.04240/
      DO 10 IPT=1,512
      Y=AK1*DAT(IPT)+A11*FM(1)+A21*FM(2)
      W=Y+2.*FM(1)+FM(2)
      FM(2)=FM(1)
      FM(1)=Y
      Y=AK2*W+A12*FM(3)+A22*FM(4)
      W=Y+2.*FM(3)+FM(4)
      FM(4)=FM(3)
      FM(3)=Y
      Y=AK3*W+A13*FM(5)+A23*FM(6)
      W=Y+2.*FM(5)+FM(6)
      FM(6)=FM(5)
      FM(5)=Y
      DAT(IPT)=W
   10 CONTINUE
      RETURN
```

```
      END
```

```
C-----MTCOMB BY BOB DICK
C-----FROM MTSORT 28 AUG 80
C-----COMB FILTERS VOICE DEPENDING ON MISTUNING
C-----DATA ARRAYS:
C-----CCR,CCI:  LIN MAG SPECTRUM, THEN COMPLEX CORRELATION
C-----DATIN:  INPUT DATA BEFORE WINDOWING
C-----IBUF:  INPUT BUFFER
C-----KBUF:  HEADER, THEN OUTPUT BUFFER
C-----OBUF:  DATA OVERLAP OUTPUT BUFFER
C-----TR,TI:  WINDOWED DATA, THEN FFT, THEN FILTERED DATA
C-----WNDW:  RAISED COSINE (HANNING) WINDOW
      BYTE F1NAM(30),F2NAM(30)
      DIMENSION DATIN(512),WNDW(512),TR(512),TI(512)
      DIMENSION CCR(512),CCI(512),OBUF(256)
      DIMENSION IBUF(256),KBUF(256),LBUF(256)
      INTEGER*4 IREC,IOSB,IBF(9),KBF(9)
      EQUIVALENCE (IBF,IBUF),(KBF,KBUF)
      DATA KBUF/256*0/DATIN/512*0./
      DATA OBUF/256*0./RAKE,COMB/-0.5,0.5/
      DATA PI,TWPI/3.14159265,6.28318531/
      DATA SAMPFQ/6400./
C-----GET INPUT FILE, NUMBER OF BLOCKS
      TYPE 10010
      ACCEPT 10020,LNTH,(F1NAM(KH),KH=1,LNTH)
      F1NAM(LNTH+1)=0
      OOPEN(UNIT=1,NAME=F1NAM,TYPE='OLD',
     1     BUFFERCOUNT=-1,ERR=9000,READONLY)
      IREC=1
      CALL READF(1,IREC,IBUF,512,IOSB)
      N1BLKS=IBF(9)
      TYPE 10030,N1BLKS
      ACCEPT 10040,NMBLKS
      IF(NMBLKS.LT.2)GO TO 9000 ----------> ABORT
      IF(NMBLKS.GT.N1BLKS)GO TO 9000 -----> ABORT
C-----GET OUTPUT FILE, WRITE HEADER
      TYPE 10050
      ACCEPT 10020,LNTH,(F2NAM(KH),KH=1,LNTH)
      F2NAM(LNTH+1)=0
      OOPEN(UNIT=2,NAME=F2NAM,TYPE='NEW',
     1     BUFFERCOUNT=-1,ERR=9000)
      KBUF(1)=1
      KBUF(2)=6400
      SEC=NMBLKS/25.
      MIN=NMBLKS/1500
      ISEC=SEC-60.*MIN
      KBUF(8)=MIN
      KBUF(9)=ISEC
      KBF(9)=NMBLKS
      CALL WRITEF(2,IREC,KBUF,512,IOSB)
C-----SET UP DATA WINDOW
      DO 10 IDX=1,512
        WNDW(IDX)=0.5-0.5*COS(PI*IDX/256.)
   10 CONTINUE
C-----GET MISTUNING TO COMB FOR
      TYPE 10060
      ACCEPT 10070,SHFFQ
      TYPE 10080
      ACCEPT 10070,PCTCMB
```

```
      CTANGL=PI*PCTCMB/100.
C-----ZERO OUTPUT BUFFER
      DO 15 IDX=1,256
        KOUF(IDX)=0
   15 CONTINUE
C-----READ FIRST DATA RECORD BEFORE MAIN LOOP.
      NMIN=1
      CALL TFSADV(DATIN,NMIN)
C-----LOOP NUMBER_OF_RECORDS-1 TIMES.
      DO 80 NMIN=2,NMBLKS
C-------MAIN LOOP.  DONE ONCE PER DATA RECORD.
        IREC=NMIN
C-------TAKE TIME WINDOWS WITH 50 PERCENT OVERLAP.
        CALL TFSADV(DATIN,NMIN)
        DO 20 IDX=1,512
          TR(IDX)=DATIN(IDX)*WNDW(IDX)
          TI(IDX)=0.
   20   CONTINUE
C-------FORM AMPLITUDE SPECTRUM
        CALL CFFT(1,TR,TI,9)
        DO 30 IDX=1,256
          CCR(IDX)=SQRT(TR(IDX)*TR(IDX)+TI(IDX)*TI(IDX))
          CCI(IDX)=0.
   30   CONTINUE
C-------SET NEGATIVE FREQUENCY PORTION TO ZERO
        DO 40 IDX=257,512
          CCR(IDX)=0.
          CCI(IDX)=0.
   40   CONTINUE
C-------TAKE INVERSE FFT
        CALL CFFT(-1,CCR,CCI,9)
C-------SEARCH FOR PEAK FROM LOW TO HIGH TIME LIMIT
        PEAK=0.
        IDPK=25
        DO 50 IDX=25,80
          TEMP=CCR(IDX)*CCR(IDX)+CCI(IDX)*CCI(IDX)
          IF(.NOT.(PEAK.LT.TEMP))GO TO 45
            PEAK=TEMP
            IDPK=IDX
   45     CONTINUE
   50   CONTINUE
C-------FIND QUADRATIC INTERPOLATION PEAK.
        BSQ=CCR(IDPK-1)*CCR(IDPK-1)+CCI(IDPK-1)*CCI(IDPK-1)
        USQ=CCR(IDPK+1)*CCR(IDPK+1)+CCI(IDPK+1)*CCI(IDPK+1)
        X=1.
        IF(PEAK.GT.USQ)X=0.
        IF(BSQ.GE.PEAK)X=-1.
        IF(X.EQ.0.)X=0.5*(USQ-BSQ)/(2.*PEAK-BSQ-USQ)
        TAU=IDPK-1+X
C-------INTERPOLATE REAL AND IMAG PARTS (VIA QUADRATIC)
        CF1=0.5*(CCR(IDPK+1)-CCR(IDPK-1))
        CF2=0.5*(CCR(IDPK-1)+CCR(IDPK+1))-CCR(IDPK)
        PARTR=CCR(IDPK)+X*(CF1+X*CF2)
        CF1=0.5*(CCI(IDPK+1)-CCI(IDPK-1))
        CF2=0.5*(CCI(IDPK-1)+CCI(IDPK+1))-CCI(IDPK)
        PARTI=CCI(IDPK)+X*(CF1+X*CF2)
        PARTM=SQRT(PARTR*PARTR+PARTI*PARTI)
        IF(PARTM.LE.0.)PARTM=1.
```

```
C-------COMPUTE PHASE ANGLES
        ANGLD=ATAN2(PARTI/PARTM,PARTR/PARTM)
        FRACA=TAU*SHFFQ/SAMPFQ
        ANGLA=TWPI*(FRACA-IFIX(FRACA))
        IF(ANGLA.GT.PI)ANGLA=ANGLA-TWPI
C-------MINUS PI.LT.ANGLA.LE.PI
        DISTA=ANGLD-ANGLA
        IF(DISTA.LT.0.)DISTA=-DISTA
        IF(DISTA.GT.PI)DISTA=TWPI-DISTA
C-------0.LE.DISTA.LE.PI
C-------TEST FOR DATA PHASE WITHIN CUT_ANGLE OF PREDICTED PHASE
        SWITCH=COMB
        IF(DISTA.GT.CTANGL)SWITCH=RAKE
C-------COMB OR RAKE ACCORDING TO PHASE DIFFERENCE
        OMEG=PI*TAU/256.
        DO 70 IDX=1,256
          CS=(PARTR*COS(OMEG*IDX)+PARTI*SIN(OMEG*IDX))/PARTM
          FILT=0.5+SWITCH*CS
          JDX=513-IDX
          TR(IDX)=FILT*TR(IDX)
          TI(IDX)=FILT*TI(IDX)
          TR(JDX)=FILT*TR(JDX)
          TI(JDX)=FILT*TI(JDX)
   70   CONTINUE
C-------TAKE INVERSE FFT
        CALL CFFT(-1,TR,TI,9)
C-------OVERLAP RESULTING TIME FUNCTIONS
        NMOUT=NMIN-1
        CALL LAPOUT(TR,OBUF,2,NMOUT)
   80 CONTINUE
C-----WRITE FINAL DATA RECORD AFTER MAIN LOOP.
      DO 90 IDX=1,256
        TR(IDX)=0.
   90 CONTINUE
      CALL LAPOUT(TR,OBUF,2,NMBLKS)
 9000 CONTINUE <------------------------ ABORT
  <-- STOP
10010 FORMAT(' MISTUNING COMBER INPUT FILE?')
10020 FORMAT(Q,30A1)
10030 FORMAT(1X,I8,' BLOCKS.  HOW MANY DO YOU WANT?')
10040 FORMAT(I8)
10050 FORMAT(' WHAT OUTPUT FILE?')
10060 FORMAT(' WHAT FREQUENCY SHIFT?(USE .)')
10070 FORMAT(F10.2)
10080 FORMAT(' WHAT PERCENT COMBING?(USE .)')
      END
```

```
      SUBROUTINE TFSADV(X,ICALL)
C-----BY BOB DICK.  REVISION OF 14 MAR 80.
C-----ADVANCES BUFFER 256 POINTS PER CALL.
      DIMENSION X(512),IBUF(256)
      INTEGER*4 IREC,IOSB
  --> IREC=ICALL+1
      CALL READF(1,IREC,IBUF,512,IOSB)
      DO 10 IDX=1,256
        X(IBX)=X(IDX+256)
        X(IDX+256)=IBUF(IDX)
   10 CONTINUE
  <-- RETURN
```

```
      END
      SUBROUTINE LAPOUT(DAT,BUF,LUN,KALL)
C-----OUTPUTS DATA WITH 50 PERCENT OVERLAP.
C-----INITIALLY BUF SHOULD BE ALL ZERO.
C-----DOES NOT INCREMENT KALL.
      DIMENSION DAT(512),BUF(256),KBUF(256)
      INTEGER*4 KREC,IOSB
      DO 10 IDX=1,256
        KBUF(IDX)=DAT(IDX)+BUF(IDX)
        BUF(IDX)=DAT(IDX+256)
   10 CONTINUE
      KREC=KALL+1
      CALL WRITEF(LUN,KREC,KBUF,512,IOSB)
      RETURN
      END
```

```
C-----PMCES BY BOB DICK
C-----FROM PTMCES 5 SEP 80
C-----ESTIMATES BOTH PITCH AND MISTUNING
C-----INCREMENTS HISTOGRAM BY CORRELATION PEAK
C-----CAN PLOT 2D PITCH-MISTUNING HISTOGRAM
C---  DATA ARRAYS:
C-----CCR,CCI:  LIN MAG SPECTRUM, THEN COMPLEX CORRELATION
C-----DATIN:  INPUT DATA BEFORE WINDOWING
C-----HMT:  HISTOGRAM OF MISTUNING GUESSES
C-----   NEG 400 TO 400 HZ BY 5 HZ
C-----HMTPT:  HISTOGRAM OF MISTUNING, PITCH
C-----   PITCH 18 BINS, 80 TO 260 HZ BY 10 HZ
C-----RESERVE 2 PTS/LINE FOR PLOTTING DATA
C-----IBUF:  INPUT BUFFER
C-----TR,TI:  WINDOWED DATA, THEN FFT
C---- WNDW:  RAISED COSINE (HANNING) WINDOW
      BYTE FINAM(30),ANSW
      DIMENSION DATIN(512),TR(512),TI(512)
      DIMENSION CCR(512),CCI(512),HMT(161)
      DIMENSION IBUF(256),HMTPT(163,18)
      INTEGER*4 IREC,IOSB,IBF(9)
      EQUIVALENCE (IBF,IBUF)
      DATA DATIN/512*0./,TWPI/6.2831853072/
      DATA HMT/161*0./,HMTPT/2934*0./
      TYPE 10010
      ACCEPT 10020,LNTH,(FINAM(KH),KH=1,LNTH)
      FINAM(LNTH+1)=0
     OOPEN(UNIT=1,NAME=FINAM,TYPE='OLD',
     1     BUFFERCOUNT=-1,ERR=9000,READONLY)
      IREC=1
      CALL READF(1,IREC,IBUF,512,IOSB)
      N1BLKS=IBF(9)
      TYPE 10030,N1BLKS
      ACCEPT 10040,NFBLK
      IF(NFBLK.LE.0)GO TO 9000 ----------> ABORT
      IF(NFBLK.GT.N1BLKS)GO TO 9000 ----------> ABORT
      ACCEPT 10040,NMBLKS
      NLBLK=NFBLK+NMBLKS-1
      IF(NLBLK.LT.NFBLK)GO TO 9000 ----------> ABORT
      IF(NLBLK.GT.N1BLKS)GO TO 9000 ----------> ABORT
      DO 10 IDX=1,512
        WNDW(IDX)=0.5-0.5*COS(3.14159265*IDX/256.)
   10 CONTINUE
      SMPK=0.
      SMPT=0.
      SMPT2=0.
      NFIRST=NFBLK+1
      LAST=NLBLK+1
      DO 80 NMIN=NFIRST,LAST
        IREC=NMIN
C-------TAKE TIME WINDOWS WITH 50 PERCENT OVERLAP.
        CALL TFSADV(DATIN,NMIN)
        DO 20 IDX=1,512
          TR(IDX)=DATIN(IDX)*WNDW(IDX)
          TI(IDX)=0.
   20   CONTINUE
        CALL CFFT(1,TR,TI,9)                    B-20
C-------FORM LINEAR MAGNITUDE SPECTRUM
```

```
      DO 30 IDX=1,256
        CCR(IDX)=SQRT(TR(IDX)*TR(IDX)+TI(IDX)*TI(IDX))
        CCI(IDX)=0.
   30 CONTINUE
C-----SET NEGATIVE FREQUENCY PORTION TO ZERO
      DO 40 IDX=257,512
        CCR(IDX)=0.
        CCI(IDX)=0.
   40 CONTINUE
C-----TAKE INVERSE FFT
      CALL CFFT(-1,CCR,CCI,9)
      PEAK=0.
C-----SEARCH FOR PEAK IN 80 TO 256 HERTZ
      DO 60 IDX=25,80
        TEMP=CCR(IDX)*CCR(IDX)+CCI(IDX)*CCI(IDX)
        IF(.NOT.(PEAK.LT.TEMP))GO TO 50
          PEAK=TEMP
          IDPK=IDX
   50   CONTINUE
   60 CONTINUE
C-----FIND QUADRATIC INTERPOLATION PEAK
      BSQ=CCR(IDPK-1)*CCR(IDPK-1)+CCI(IDPK-1)*CCI(IDPK-1)
      USQ=CCR(IDPK+1)*CCR(IDPK+1)+CCI(IDPK+1)*CCI(IDPK+1)
      X=1.
      IF(PEAK.GT.USQ)X=0.
      IF(BSQ.GE.PEAK)X=-1.
      IF(X.EQ.0.)X=0.5*(USQ-BSQ)/(2.*PEAK-BSQ-USQ)
      TAU=IDPK-1+X
C-----INTERPOLATE REAL AND IMAG PARTS
      CF1=0.5*(CCR(IDPK+1)-CCR(IDPK-1))
      CF2=0.5*(CCR(IDPK-1)+CCR(IDPK+1))-CCR(IDPK)
      PARTR=CCR(IDPK)+X*(CF1+X*CF2)
      CF1=0.5*(CCI(IDPK+1)-CCI(IDPK-1))
      CF2=0.5*(CCI(IDPK-1)+CCI(IDPK+1))-CCI(IDPK)
      PARTI=CCI(IDPK)+X*(CF1+X*CF2)
      PARTM=SQRT(PARTR*PARTR+PARTI*PARTI)
      IF(PARTM.LE.0.)PARTM=1.
C-----FORM INTERPOLATED PEAK FROM PARTM.
      PEAK=PARTM*PARTM
      SMPK=SMPK+PEAK
      ANGL=ATAN2(PARTI/PARTM,PARTR/PARTM)
      PITCH=6400./TAU
      SMPT=SMPT+PEAK*PITCH
      SMPT2=SMPT2+PEAK*PITCH*PITCH
      SHF=PITCH*ANGL/TWPI
      IXPT=(PITCH-70.)/10.
      IF(IXPT.LT.1)IXPT=1
      IF(IXPT.GT.18)IXPT=18
      GO TO 68
   65 CONTINUE
        SHF=SHF-PITCH
   68 CONTINUE
      IF(SHF.GE.-402.)GO TO 65
      DO 70 IDX=1,11
        SHF=SHF+PITCH
C-------ROUND INSTEAD OF TRUNCATING
        IXMT=81.5+SHF/5.
C-----IF(POINTER OUT OF RANGE)EXIT LOOP
```

```
            IF(.NOT.(IXMT.LE.161))GO TO 75
              HMT(IXMT)=HMT(IXMT)+PEAK
              HMTPT(IXMT,IXPT)=HMTPT(IXMT,IXPT)+PEAK
 70         CONTINUE
 75         CONTINUE
 80       CONTINUE
          AVPT=SMPT/SMPK
          SDPT=SQRT(SMPT2/SMPK-AVPT*AVPT)
          TYPE 10050,AVPT,SDPT
          PKMT=0.
          MTX=81
          DO 100 IDX=1,161
            IF(.NOT.(PKMT.LT.HMT(IDX)))GO TO 90
              PKMT=HMT(IDX)
              MTX=IDX
 90         CONTINUE
100       CONTINUE
          FRMT=(MTX-81)*5.
          TYPE 10060,FRMT
          TYPE 10070
          ACCEPT 10080,ANSW
          IF(.NOT.(ANSW.EQ.'Y'))GO TO 110
            CALL PLTHST(HMTPT)
            ACCEPT 10080,ANSW
110       CONTINUE
9000      CONTINUE <---------- ABORT
          STOP
10010     FORMAT(' PITCH-MISTUNING ESTIMATOR INPUT FILE?')
10020     FORMAT(Q,30A1)
10030     FORMAT(1X,I8,' BLOCKS.  GIVE FIRST <CR> HOW MANY.')
10040     FORMAT(I8)
10050     FORMAT(' PITCH AVG, STD DEV:',2F8.1)
10060     FORMAT(' ESTIMATED MISTUNING (HZ):',F9.0)
10070     FORMAT(' DO YOU WANT A GRAPH?(Y FOR YES)')
10080     FORMAT(A1)
          END
-------------------------------------------------------------------
          SUBROUTINE TFSADV(X,ICALL)
C-----BY BOB DICK,  REVISION OF 14 MAR 80.
C-----ADVANCES BUFFER 256 POINTS PER CALL.
          DIMENSION X(512),IBUF(256)
          INTEGER*4 IREC,IOSB
          IREC=ICALL+1
          CALL READF(1,IREC,IBUF,512,IOSB)
          DO 10 IDX=1,256
            X(IDX)=X(IDX+256)
            X(IDX+256)=IBUF(IDX)
 10       CONTINUE
          RETURN
          END
-------------------------------------------------------------------
          SUBROUTINE PLTHST(HMTPT)
C-----BY BOB DICK 4 AUG 80
C-----LAST UPDATE 3 SEP 80
C-----PLOTS 2D HISTOGRAM OF MISTUNING VS PITCH
          DIMENSION HMTPT(163,18),XCRD(163)
          DIMENSION AXISX(4),AXISY(4)
          DATA AXISX/81.,81.,1.,13./AXISY/0.,9.2,0.,1./
C-----FIND HISTOGRAM MAXIMUM
          HMAX=0.
```

```
      DO 20 IDX2=1,18
      DO 10 IDX1=1,161
      TEMP=HMTPT(IDX1,IDX2)
      IF(HMAX.LT.TEMP)HMAX=TEMP
   10 CONTINUE
   20 CONTINUE
C-----GENERATE X COORDINATES
      DO 30 IDX1=1,161
      XCRD(IDX1)=IDX1
   30 CONTINUE
      XCRD(162)=1.
C-----160 INTERVALS AT 13/IN IS 12.3 IN HORIZ
      XCRD(163)=13.
C-----VERT SCALE IS HIST MAX PER 0.8 INCH
      VSCALE=HMAX/0.8
C-----CLEAR SCREEN, RESET ORIGIN
      CALL PLOTS
      DO 40 LNE=1,18
C-----DONE ONCE PER LINE
C-----17 INTERVALS AT 0.5 INCH EACH IS 8.5 IN VERT
      YORG=(18-LNE)*0.5
      CALL PLOT(0.,YORG,-3)
      HMTPT(162,LNE)=0.
      HMTPT(163,LNE)=VSCALE
C-----PLOT LINE
      CALL LINE(XCRD,HMTPT(1,LNE),161,1,0,0)
      YORG=-YORG
      CALL PLOT(0.,YORG,-3)
   40 CONTINUE
C-----PLOT 0 HZ MISTUNING AXIS
      CALL LINE(AXISX,AXISY,2,1,0,0)
C-----REPEAT LAST PLOT LINE TO FILL BUFFER
      CALL LINE(XCRD,HMTPT(1,18),161,1,0,0)
      RETURN
      END
```

```
C-----VCRFSP BY BOB DICK 29 MAY 80.  LAST UPDATE 30 MAY 80.
C-----FROM VCRFPL 29 MAY 80.
C-----VOICE RE-FILTER AND SEPARATE.
C-----LATTICE FILTER, THRESHOLD RESIDUAL, RECONSTRUCT
C-----DATA ARRAYS:
C-----BRSD:  BACKWARD RESIDUAL
C-----COEF:  LATTICE FILTER COEFFICIENTS
C-----FRSD:  FORWARD RESIDUAL
C-----RFCOEF:  RECONSTRUCTION FILTER COEFFICIENTS
C-----SIGDIF:  INPUT SIGNAL AFTER DIFFERENCING
C-----SIGIN:  INPUT SIGNAL
C-----SIGOUT:  OUTPUT (RECONSTRUCTED) SIGNAL
C-----TRSD:  FORWARD RESIDUAL AFTER THRESHOLDING
      DIMENSION BRSD(150),COEF(20),FRSD(150),RFCOEF(20)
      DIMENSION SIGDIF(150),SIGIN(150),SIGOUT(150),TRSD(150)
      DIMENSION BLANKS(64),SIGSEC(150)
      INTEGER*4 IREC,IOSB,IBF(9),KBF(9)
      BYTE F1NAM(30),F2NAM(30),F3NAM(30)
      DIMENSION IBUF(256),KBUF(256)
      EQUIVALENCE (IBF,IBUF),(KBF,KBUF)
      DATA KBUF/256*0/BLANKS/64*0./
      TYPE 10010
      ACCEPT 10020,LNTH,(F1NAM(KH),KH=1,LNTH)
      F1NAM(LNTH+1)=0
     OOPEN(UNIT=1,NAME=F1NAM,TYPE='OLD',
     1     BUFFERCOUNT=-1,ERR=9000,READONLY)
      IREC=1
      CALL READF(1,IREC,IBUF,512,IOSB)
      N1BLKS=IBF(9)
      TYPE 10030,N1BLKS
      ACCEPT 10040,NMBLKS
      IF(NMBLKS.LE.0)NMBLKS=N1BLKS
      IF(NMBLKS.GE.N1BLKS)NMBLKS=N1BLKS-1
      NMLPS=NMBLKS*2
      TYPE 10050
      ACCEPT 10020,LNTH,(F2NAM(KH),KH=1,LNTH)
      F2NAM(LNTH+1)=0
     OOPEN(UNIT=2,NAME=F2NAM,TYPE='NEW',
     1     BUFFERCOUNT=-1,ERR=9000)
      ACCEPT 10020,LNTH,(F3NAM(KH),KH=1,LNTH)
      F3NAM(LNTH+1)=0
     OOPEN(UNIT=8,NAME=F3NAM,TYPE='NEW',
     1     BUFFERCOUNT=-1,ERR=9000)
      KBUF(1)=1
      KBUF(2)=6400
      SEC=NMBLKS/25.
      MIN=NMBLKS/1500
      ISEC=SEC-60.*MIN
      KBUF(8)=MIN
      KBUF(9)=ISEC
      KBF(9)=NMBLKS
      CALL WRITEF(2,IREC,KBUF,512,IOSB)
      CALL WRITEF(8,IREC,KBUF,512,IOSB)
      TYPE 10060
      ACCEPT 10070,RSTH
      CALL INTGD(148,20)
      CALL WROUT(BLANKS,20,1,2)
      CALL WROUT(BLANKS,20,2,8)
```

```
C-----THIS LINES UP OUTPUTS WITH INPUTS.
      DO 60 KLPS=1,NMLPS
      |  CALL GETDAT(1,SIGIN,ICODE)
C-----|--ICODE NEGATIVE MEANS NO MORE DATA
      IF(.NOT.(ICODE.GE.0))GO TO 70
C-----|--DIFFERENCE SIGNAL FOR PRE-EMPHASIS
      |  SIGDIF(1)=SIGIN(1)
      |  DO 20 IDX=2,148
      |  |  SIGDIF(IDX)=SIGIN(IDX)-SIGIN(IDX-1)
   20 |  CONTINUE
      |  CALL LTFLT(SIGDIF,148,COEF,20,FRSD,BRSD)
      |  CALL BLNKR(FRSD,TRSD,148,RSTH)
      |  CALL RLTFLT(TRSD,148,COEF,20,SIGOUT)
C-----|--SUM SIGNAL TO UNDO PRE-EMPHASIS
      |  DO 30 IDX=2,148
      |  |  SIGOUT(IDX)=SIGOUT(IDX)+SIGOUT(IDX-1)
   30 |  CONTINUE
      |  CALL WROUT(SIGOUT(21),128,1,2)
      |  DO 50 IDX=21,148
      |  |  SIGSEC(IDX)=SIGIN(IDX)-SIGOUT(IDX)
   50 |  CONTINUE
      |  CALL WROUT(SIGSEC(21),128,2,8)
   60 CONTINUE
   70 CONTINUE
 9000 CONTINUE
      STOP
10010 FORMAT(' INPUT FILE FOR VOICE RECONSTRUCTOR?')
10020 FORMAT(Q,30A1)
10030 FORMAT(1X,I8,' BLOCKS.  HOW MANY DO YOU WANT?')
10040 FORMAT(I8)
10050 FORMAT(' WHAT TWO OUTPUT FILES?')
10060 FORMAT(' MULT OF STD DEV FOR RESID THRESH?(USE .)')
10070 FORMAT(F12.5)
      END
```

```
      SUBROUTINE GETDAT(LUN,RDAT,ICODE)
C-----BY BOB DICK 1 MAY 80.
C-----READS OVERLAPPING DATA BLOCKS.
C-----IBUF IS INPUT BUFFER, POINTER KIB
C-----DBUF IS DATA BUFFER, POINTER KDB
C-----RDAT IS DATA RETURNED
C-----NSIZ,NISZ IS DATA SIZE PER CALL
C-----NVLP,NVLP IS DATA OVERLAP BETWEEN CALLS
C-----INTGD IS ENTRY TO INITIALIZE AND SET
C----------SIZE AND OVERLAP.
      DIMENSION IBUF(256),DBUF(1024),RDAT(1024),IOSB(2)
      BYTE ERRBYT
      EQUIVALENCE (ERRBYT,IOSB)
      INTEGER*4 KREC
      NFIRST=KDB+1
      DO 30 KDB=NFIRST,NSIZ
      |  IF(.NOT.(KIB.EQ.0))GO TO 10
C-----|--HERE NEED MORE DATA
      |  |  KREC=KREC+1
      |  |  CALL READF(LUN,KREC,IBUF,512,IOSB)
      |  |  KIB=-1
      |  |  IF(ERRBYT.GE.0)KIB=1
   10 |  CONTINUE
      |  IF(.NOT.(KIB.GT.0))GO TO 20
      |
```

```
C--------HERE THERE IS NO END OF FILE
      DBUF(KDB)=IBUF(KIB)
      KIB=KIB+1
      IF(KIB.GT.256)KIB=0
   20 CONTINUE
   30 CONTINUE
      IF(.NOT.(KIB.GE.0))GO TO 70
C-----HERE NO EOF
      DO 40 KRD=1,MSIZ
      RDAT(KRD)=DBUF(KRD)
   40 CONTINUE
      IF(.NOT.(MVLP.GT.0))GO TO 60
C--------HERE THERE IS OVERLAP
      DO 50 KDB=1,MVLP
      DBUF(KDB)=DBUF(KDB+MSIZ-MVLP)
   50 CONTINUE
   60 CONTINUE
      KDB=MVLP
   70 CONTINUE
      ICODE=KIB
      IF(ICODE.GT.0)ICODE=0
      RETURN
      ENTRY INTGD(NSIZ,NVLP)
C-----INITIAL ENTRY
      MSIZ=1
      IF(1.LE.NSIZ.AND.NSIZ.LE.1024)MSIZ=NSIZ
      MVLP=0
      IF(0.LE.NVLP.AND.NVLP.LT.NSIZ)MVLP=NVLP
C-----SKIP HEADER BLOCK
      KREC=1
      KIB=0
      KDB=0
      RETURN
      END
```

```
      SUBROUTINE LTFLT(SIG,LEN,COEF,NORDER,FRSD,BRSD)
C-----BY BOB DICK 1 MAY 80.  LAST UPDATE 2 MAY 80
C-----NEW MORE GENERAL VERSION OF ADFLT.
C-----SIG IS INPUT WAVEFORM LENGTH LEN
C-----COEF WILL BE LATTICE FILTER COEFS, NORDER OF THEM
C-----FRSD WILL BE FORWARD RESIDUAL
C-----BRSD WILL BE BACKWARD RESIDUAL (FIRST NORDER ZEROED)
C-----DOES NOT ITSELF DIFFERENCE SIGNAL FOR PRE-EMPHASIS
      DIMENSION SIG(1024),COEF(32),FRSD(1024),BRSD(1024)
      IF(.NOT.(1.LE.NORDER.AND.NORDER.LE.32))GO TO 9000    --> ABORT
      IF(.NOT.(NORDER+1.LE.LEN.AND.LEN.LE.1024))GO TO 9000 --> ABORT
      FRSD(1)=SIG(1)
      BRSD(1)=0.
      DO 10 IDX=2,LEN
      FRSD(IDX)=SIG(IDX)
      BRSD(IDX)=SIG(IDX-1)
   10 CONTINUE
      DO 40 ITER=1,NORDER
      SFSQ=0.
      SBSQ=0.
      DOTPR=0.
      DO 20 IDX=ITER+1,LEN
      SFSQ=SFSQ+FRSD(IDX)*FRSD(IDX)
      SBSQ=SBSQ+BRSD(IDX)*BRSD(IDX)
```

```
            DOTPR=DOTPR+FRSD(IDX)*BRSD(IDX)
   20     CONTINUE
          DENOM=SFSQ+SBSQ
          IF(DENOM.LE.0.)DENOM=1.
          COEF(ITER)=-2.*DOTPR/DENOM
          BROLD=0.
          DO 30 IDX=ITER+1,LEN
            BRTMP=COEF(ITER)*FRSD(IDX)+BRSD(IDX)
            FRSD(IDX)=FRSD(IDX)+COEF(ITER)*BRSD(IDX)
            BRSD(IDX)=BROLD
            BROLD=BRTMP
   30     CONTINUE
   40   CONTINUE
 9000   CONTINUE <-------------- ABORT
        RETURN
        END
```

```
        SUBROUTINE BLNKR(DATIN,DATOUT,LEN,THRES)
C-----BY BOB DICK 1 MAY 80
C-----DATOUT=DATIN WITH SMALL VALUES ZEROED.
C-----VALUE IS SMALL IF WITHIN STD DEV*THRES OF ZERO
C-----DATIN CAN ALSO BE DATOUT.
        DIMENSION DATIN(1024),DATOUT(1024)
        IF(.NOT.(1.LE.LEN.AND.LEN.LE.1024))GO TO 9000 ----> ABORT
        SUMSQ=0.
        DO 10 IDX=1,LEN
          SUMSQ=SUMSQ+DATIN(IDX)*DATIN(IDX)
   10   CONTINUE
        STDV=SQRT(SUMSQ/LEN)
        CUTF=THRES*STDV
        DO 20 IDX=1,LEN
          RESULT=0.
          IF(DATIN(IDX).LE.-CUTF)RESULT=DATIN(IDX)
          IF(CUTF.LE.DATIN(IDX))RESULT=DATIN(IDX)
          DATOUT(IDX)=RESULT
   20   CONTINUE
 9000   CONTINUE <-------------- ABORT
        RETURN
        END
```

```
        SUBROUTINE RLTFLT(DAT,LEN,COEF,NORDER,RES)
C-----BY BOB DICK 2 MAY 80
C-----DOES RECIPROCAL OF LATTICE FILTERING
C-----DAT IS INPUT (APPROX OF LATTICE FORWARD RESIDUAL)
C-----LEN IS LENGTH OF INPUT AND OUTPUT
C-----COEF IS VECTOR OF LATTICE FILTER COEFFICIENTS
C-----NORDER IS NUMBER OF FILTER COEFFICIENTS
C-----RES IS OUTPUT
C-----BRSD IS REPRODUCTION OF VECTOR OF BACKWARD RESIDUALS
C-----DOES NOT SUM OUTPUT TO UNDO PRE-EMPHASIS
        DIMENSION DAT(1024),COEF(32),RES(1024),BRSD(33)
        IF(.NOT.(1.LE.NORDER.AND.NORDER.LE.32))GO TO 9000 ----> ABORT
        IF(.NOT.(NORDER+1.LE.LEN.AND.LEN.LE.1024))GO TO 9000 ----> ABORT
        DO 30 KDAT=1,LEN
          FWEX=DAT(KDAT)
          IF(.NOT.(KDAT.GE.2))GO TO 20
            NSTG=KDAT-1
            IF(NSTG.GT.NORDER)NSTG=NORDER
            DO 10 JSTG=1,NSTG
              KSTG=NSTG+1-JSTG
```

```
            FPRE=FNEX-COEF(KSTG)*BRSD(KSTG)
            BRSD(KSTG+1)=COEF(KSTG)*FPRE+BRSD(KSTG)
            FNEX=FPRE
   10     CONTINUE
   20     CONTINUE
          RES(KDAT)=FNEX
          BRSD(1)=FNEX
   30   CONTINUE
 9000   CONTINUE<-------------ABORT
        RETURN
          END
-------------------------------------------------------------------
          SUBROUTINE WROUT(DAT,NUM,ICHAN,LUN)
C-----BY BOB DICK 29 MAY 80.  LAST UPDATE 30 MAY 80.
C-----MULTICHANNEL OUTPUT BUFFERER AND WRITER
C-----DAT IS OUTPUT DATA
C-----NUM IS NUMBER OF POINTS
C-----ICHAN TELLS WHICH OF 4 CHANNELS
C-----LUN IS OUTPUT LOGICAL UNIT NUMBER
C-----INWRT IS RE-INITIALIZATION ENTRY
          INTEGER*4 IOSB,KREC(4)
          DIMENSION DAT(768),IBUF(1024,4),KBUF(4)
          DATA KBUF,KREC/4*0,4*2/
        IF(.NOT.(1.LE.NUM.AND.NUM.LE.1024))GO TO 9000------>ABORT 1
          IF(.NOT.(1.LE.ICHAN.AND.ICHAN.LE.4))GO TO 9000------>ABORT 1
          LBUF=KBUF(ICHAN)
          DO 10 KDAT=1,NUM
            LBUF=LBUF+1
            IBUF(LBUF,ICHAN)=DAT(KDAT)
   10     CONTINUE
   20     CONTINUE
          IF(.NOT.(LBUF.GE.256))GO TO 40
            CALL WRITEF(LUN,KREC(ICHAN),IBUF(1,ICHAN),512,IOSB)
            KREC(ICHAN)=KREC(ICHAN)+1
            DO 30 IDX=1,768
              IBUF(IDX,ICHAN)=IBUF(IDX+256,ICHAN)
   30       CONTINUE
            LBUF=LBUF-256
          GO TO 20
   40     CONTINUE
          KBUF(ICHAN)=LBUF
 9000   CONTINUE <-------------ABORT 1
        RETURN
          ENTRY INWRT(ICHAN)
        IF(.NOT.(1.LE.ICHAN.AND.ICHAN.LE.4))GO TO 9100------>ABORT 2
          KBUF(ICHAN)=0
          KREC(ICHAN)=2
 9100   CONTINUE<-------------ABORT 2
        RETURN
          END
```

```
C-----MISTUNED SSB MAIN.  LAST UPDATE 6 OCT 80
C-----WRITTEN 21 MAR 80
C-----DATA ARRAYS:
C-----DBLR,DBLI:  DOUBLED SAMPLE RATE REAL, IMAG PARTS
C-----FRE,FIM:  FUTURE DATA REAL, IMAG PARTS
C-----FMR,FMI:  LOWPASS FILTER MEMORIES FOR REAL, IMAG DATA
C-----FMO:  LOWPASS FILTER MEMORY FOR OUTPUT DATA
C-----FRES:  HILBERT FILTER FUTURE RESULT (TRAILER)
C-----IBUF:  INPUT-OUTPUT BUFFER
C-----OLRE,OLIM:  OLD DATA REAL, IMAG PARTS
C-----PRES:  HILBERT FILTER PREVIOUS RESULT (LEADER)
C-----REO:  (REAL) OUTPUT DATA
      BYTE GNAME(30),FNAME(30)
      INTEGER*4 IREC,IOSB,IB(9)
      DIMENSION IBUF(256),FRE(256),FIM(256)
      DIMENSION PRES(15),RES(256),FRES(15)
      DIMENSION OLRE(256),OLIM(256),REO(256)
      DIMENSION FMR(6),FMI(6),FMO(6),DBLR(512),DBLI(512)
      EQUIVALENCE (IB,IBUF)
      DATA FRE,FIM,FRES/256*0.,256*0.,15*0./
      DATA FMR,FMI,FMO/6*0.,6*0.,6*0./
C-----GET DATA FILE NAME
      TYPE 10000
      ACCEPT 10010,LNTH,(FNAME(KH),KH=1,LNTH)
      FNAME(LNTH+1)=0
C-----OPEN WAVEFORM FILE, READ SIZE.
      OPEN(UNIT=2,NAME=FNAME,TYPE='OLD',
     2     BUFFERCOUNT=-1,ERR=9000,READONLY)
      IREC=1
      CALL READF(2,IREC,IBUF,512,IOSB)
      NMBLKS=IB(9)
      TYPE 10013,NMBLKS
C-----GET NUMBER OF BLOCKS TO PROCESS.
      ACCEPT 10017,NBLK
      IF(.NOT.(NBLK.GE.1))GO TO 9000 ----------> ABORT
      IF(.NOT.(NBLK.LE.NMBLKS))GO TO 9000 ----------> ABORT
C-----GET MISTUNING TO SIMULATE.
      TYPE 10020
      ACCEPT 10030,FQSF
      OMEGA=3.141592654*FQSF/6400.
      ANGLE=0.
C-----OPEN OUTPUT FILE, WRITE HEADER.
      TYPE 10040
      ACCEPT 10010,LNTH,(GNAME(KH),KH=1,LNTH)
      GNAME(LNTH+1)=0
      OPEN(UNIT=1,NAME=GNAME,TYPE='NEW',
     2     BUFFERCOUNT=-1,ERR=9000)
      DO 10 IDX=1,256
        IBUF(IDX)=0
   10 CONTINUE
      IBUF(1)=1
      IBUF(2)=6400
      SEC=NBLK/25.
      MIN=NBLK/1500
      IBUF(8)=MIN
      IBUF(9)=SEC-60.*MIN
      IB(9)=NBLK
      CALL WRITEF(1,IREC,IBUF,512,IOSB)
```

```
      TWPI=2.*3.141592654
C-----LOOP NUMBER_OF_BLOCKS+1 TIMES.
      LAST=NBLK+2
      DO 150 JBLK=2,LAST
C-----IF(NOT LAST TIME)GET DATA.  ELSE USE ZEROES.
      IF(.NOT.(JBLK.LT.LAST))GO TO 30
C-----HERE MORE DATA.  USE IT.
      IREC=JBLK
      CALL READF(2,IREC,IBUF,512,IOSB)
      GO TO 50
   30 CONTINUE
C-----HERE NO MORE DATA.  USE ZEROES.
      DO 40 IDX=1,256
         IBUF(IDX)=0
   40 CONTINUE
   50 CONTINUE
C-----AGE THE DATA (OLD:=FUTURE; FUTURE:=NEW)
      DO 60 IDX=1,256
      OLRE(IDX)=FRE(IDX)
      OLIM(IDX)=FIM(IDX)
      FRE(IDX)=IBUF(IDX)
      FIM(IDX)=0.
   60 CONTINUE
C-----SAVE PREVIOUS HILBERT FILTERING TRAILER.
      DO 70 IDX=1,15
      FIM(IDX)=-FRES(IDX)
   70 CONTINUE
C-----DO HILBERT FILTERING.
      CALL HILB(FRE,PRES,RES,FRES)
      DO 80 IDX=1,256
      FIM(IDX)=FIM(IDX)-RES(IDX)
   80 CONTINUE
C-----UPDATE PREDICTION (HILBERT FILTERING LEADER).
      DO 90 IDX=1,15
      OLIM(IDX+241)=OLIM(IDX+241)-PRES(IDX)
   90 CONTINUE
C-----DO TWO INPUTS BEFORE FIRST OUTPUT.
C-----IF(FIRST TIME THRU)SKIP OUTPUT.
      IF(.NOT.(JBLK.GT.2))GO TO 140
C-----HERE NOT FIRST BLOCK.  PRODUCE OUTPUT.
C-----DOUBLE SAMPLING RATE BY INTERPOLATING ZEROES.
      DO 100 IDX=1,256
      DBLR(IDX*2-1)=OLRE(IDX)
      DBLR(IDX*2)=0.
      DBLI(IDX*2-1)=OLIM(IDX)
      DBLI(IDX*2)=0.
  100 CONTINUE
C-----SMOOTH VIA LOWPASS FILTER.
      CALL LWPS(FMR,DBLR)
      CALL LWPS(FMI,DBLI)
C-----MULTIPLY BY COMPLEX EXPONENTIAL, KEEP REAL PART.
      DO 110 IDX=1,512
      ANGLE=ANGLE+OMEGA
      IF(ANGLE.GT.TWPI)ANGLE=ANGLE-TWPI
      IF(ANGLE.LT.-TWPI)ANGLE=ANGLE+TWPI
      DBLR(IDX)=DBLR(IDX)*COS(ANGLE)-
                DBLI(IDX)*SIN(ANGLE)
  110 CONTINUE
```

```
C----------LOWPASS TO PREVENT ALIASING.
         CALL LWPS(FNO,DBLR)
C----------HALVE THE SAMPLING RATE.
         DO 120 IDX=1,256
           REO(IDX)=DBLR(IDX*2)
  120    CONTINUE
C----------WRITE RESULT.
         DO 130 IDX=1,256
C----------MULT BY 2 TO COMPENSATE FOR ATTEN
           IBUF(IDX)=2.*REO(IDX)
  130    CONTINUE
         IREC=JBLK-1
         CALL WRITEF(1,IREC,IBUF,512,IOSB)
  140  CONTINUE
  150 CONTINUE
      CLOSE(UNIT=2)
      CLOSE(UNIT=1)
      TYPE 10050
 9000 CONTINUE <---------- ABORT
 <--- STOP
10000 FORMAT(' FILE NAME FOR SSB SIMULATION?')
10010 FORMAT(Q,30A1)
10013 FORMAT(' HAVE',I8,' BLOCKS.  HOW MANY DO YOU WANT?')
10017 FORMAT(I8)
10020 FORMAT(' FREQUENCY SHIFT OF MISTUNING?(HZ)')
10030 FORMAT(F8.0)
10040 FORMAT(' FILE NAME FOR OUTPUT?')
10050 FORMAT(' FILE WRITTEN.')
      END
```

```
C-----WTMERG BY BOB DICK 11 MAR 80.  UPDATED 15 APR 80.
C-----DOES WEIGHTED MERGE OF TWO FILES.
      BYTE F1NAM(30),F2NAM(30),F3NAM(30)
      DIMENSION IBUF(256),JBUF(256),KBUF(256)
      INTEGER*4 IREC,IOSB,IDBF(9),JDBF(9),KDBF(9)
      EQUIVALENCE (IDBF,IBUF),(JDBF,JBUF),(KDBF,KBUF)
      DATA F1NAM/30*0/F2NAM/30*0/F3NAM/30*0/
      DATA KBUF/256*0/
      TYPE 10010
      ACCEPT 10020,F1NAM
      F1NAM(30)=0
      OOPEN(UNIT=1,NAME=F1NAM,TYPE='OLD',
     1     BUFFERCOUNT=-1,ERR=9000,READONLY)
      ACCEPT 10020,F2NAM
      F2NAM(30)=0
      OOPEN(UNIT=2,NAME=F2NAM,TYPE='OLD',
     1     BUFFERCOUNT=-1,ERR=9000,READONLY)
      IREC=1
      CALL READF(1,IREC,IBUF,512,IOSB)
      CALL READF(2,IREC,JBUF,512,IOSB)
      N1BLKS=IDBF(9)
      N2BLKS=JDBF(9)
      NSBLKS=N1BLKS
      IF(NSBLKS.GT.N2BLKS)NSBLKS=N2BLKS
      TYPE 10030,NSBLKS
      ACCEPT 10040,NOBLKS
      IF(NOBLKS.LE.0)NOBLKS=NSBLKS
      IF(NOBLKS.GT.NSBLKS)NOBLKS=NSBLKS
      RMS1=0.
      RMS2=0.
      DO 20 IBLK=1,NOBLKS
        IREC=IBLK+1
        CALL READF(1,IREC,IBUF,512,IOSB)
        CALL READF(2,IREC,JBUF,512,IOSB)
        BSUM1=0.
        BSUM2=0.
        DO 10 IDX=1,256
          X=IBUF(IDX)
          BSUM1=BSUM1+X*X
          X=JBUF(IDX)
          BSUM2=BSUM2+X*X
   10   CONTINUE
        RMS1=RMS1+BSUM1/256.
        RMS2=RMS2+BSUM2/256.
   20 CONTINUE
      RMS1=SQRT(RMS1/NOBLKS)
      RMS2=SQRT(RMS2/NOBLKS)
      DB2=20.*ALOG(RMS2/RMS1)/ALOG(10.)
      TYPE 10050,RMS1,RMS2,DB2
      TYPE 10060
      ACCEPT 10070,GNDB
      TYPE 10080
      ACCEPT 10020,F3NAM
      F3NAM(30)=0
      IF(.NOT.(F3NAM(1).NE.' '))GO TO 9000 ----> ABORT
      OOPEN(UNIT=8,NAME=F3NAM,TYPE='NEW',
     1     BUFFERCOUNT=-1,ERR=9000)
      KBUF(1)=1
```

```
      KBUF(2)=6400
      SEC=NOBLKS/25.
      MIN=NOBLKS/1500
      ISEC=SEC-60.*MIN
      KBUF(8)=MIN
      KBUF(9)=ISEC
      KDBF(9)=NOBLKS
      IREC=1
      CALL WRITEF(8,IREC,KBUF,512,IOSB)
      GAIN=10.**(GNDB/20.)
      FCTR1=1./(1.+GAIN)
      FCTR2=GAIN*FCTR1
      DO 40 IBLK=1,NOBLKS
        IREC=IBLK+1
        CALL READF(1,IREC,IBUF,512,IOSB)
        CALL READF(2,IREC,JBUF,512,IOSB)
        DO 30 IDX=1,256
          KBUF(IDX)=FCTR1*IBUF(IDX)+FCTR2*JBUF(IDX)
   30   CONTINUE
        CALL WRITEF(8,IREC,KBUF,512,IOSB)
   40 CONTINUE
      TYPE 10090,NOBLKS
 9000 CONTINUE <---------- ABORT
      STOP
10010 FORMAT(' TWO FILES FOR WEIGHTED MERGE?')
10020 FORMAT(30A1)
10030 FORMAT(' HAVE',I8,' BLOCKS.  HOW MANY TO MERGE?')
10040 FORMAT(I8)
10050 FORMAT(' RMS',2F10.2,', 2ND/1ST:',F10.2,' DB.')
10060 FORMAT(' WHAT DB GAIN FOR 2ND FILE?')
10070 FORMAT(F10.2)
10080 FORMAT(' WHAT OUTPUT FILE?')
10090 FORMAT(' HEADER AND',I8,' DATA BLOCKS WRITTEN.')
      END
```

```
C-----WTMMRG BY BOB DICK
C-----FROM WTMERG 29 AUG 80
C-----DOES WEIGHTED MERGE OF TWO FILES AND ADDS GAUSSIAN NOISE.
      BYTE F1NAM(30),F2NAM(30),F3NAM(30)
      DIMENSION IBUF(256),JBUF(256),KBUF(256)
      INTEGER*4 IREC,IOSB,IDBF(9),JDBF(9),KDBF(9)
      EQUIVALENCE (IDBF,IBUF),(JDBF,JBUF),(KDBF,KBUF)
      DATA F1NAM/30*0/F2NAM/30*0/F3NAM/30*0/
      DATA KBUF/256*0/
      TYPE 10010
      ACCEPT 10020,F1NAM
      F1NAM(30)=0
      OOPEN(UNIT=1,NAME=F1NAM,TYPE='OLD',
     1     BUFFERCOUNT=-1,ERR=9000,READONLY)
      ACCEPT 10020,F2NAM
      F2NAM(30)=0
      OOPEN(UNIT=2,NAME=F2NAM,TYPE='OLD',
     1     BUFFERCOUNT=-1,ERR=9000,READONLY)
      IREC=1
      CALL READF(1,IREC,IBUF,512,IOSB)
      CALL READF(2,IREC,JBUF,512,IOSB)
      N1BLKS=IDBF(9)
      N2BLKS=JDBF(9)
      NSBLKS=N1BLKS
      IF(NSBLKS.GT.N2BLKS)NSBLKS=N2BLKS
      TYPE 10030,NSBLKS
      ACCEPT 10040,NOBLKS
      IF(NOBLKS.LE.0)NOBLKS=NSBLKS
      IF(NOBLKS.GT.NSBLKS)NOBLKS=NSBLKS
      RMS1=0.
      RMS2=0.
      DO 20 IBLK=1,NOBLKS
        IREC=IBLK+1
        CALL READF(1,IREC,IBUF,512,IOSB)
        CALL READF(2,IREC,JBUF,512,IOSB)
        BSUM1=0.
        BSUM2=0.
        DO 10 IDX=1,256
          X=IBUF(IDX)
          BSUM1=BSUM1+X*X
          X=JBUF(IDX)
          BSUM2=BSUM2+X*X
10      CONTINUE
        RMS1=RMS1+BSUM1/256.
        RMS2=RMS2+BSUM2/256.
20    CONTINUE
      RMS1=SQRT(RMS1/NOBLKS)
      RMS2=SQRT(RMS2/NOBLKS)
      DB2=20.*LOG(RMS2/RMS1)/LOG(10.)
      TYPE 10050,RMS1,RMS2,DB2
      TYPE 10060
      ACCEPT 10070,GNDB
      TYPE 10075
      ACCEPT 10070,BBNS
      TYPE 10080
      ACCEPT 10020,F3NAM
      F3NAM(30)=0
      IF(.NOT.(F3NAM(1).NE.' '))GO TO 9000 ----------> ABORT
```

```
C-----WTMMRG BY BOB BICK
C-----FROM WTMERG 29 AUG 80
C-----DOES WEIGHTED MERGE OF TWO FILES AND ADDS GAUSSIAN NOISE.
      BYTE F1NAM(30),F2NAM(30),F3NAM(30)
      DIMENSION IBUF(256),JBUF(256),KBUF(256)
      INTEGER*4 IREC,IOSB,IDBF(9),JBBF(9),KBBF(9)
      EQUIVALENCE (IDBF,IBUF),(JDBF,JBUF),(KDBF,KBUF)
      DATA F1NAM/30*0/F2NAM/30*0/F3NAM/30*0/
      DATA KBUF/256*0/
      TYPE 10010
      ACCEPT 10020,F1NAM
      F1NAM(30)=0
     OOPEN(UNIT=1,NAME=F1NAM,TYPE='OLD',
     1     BUFFERCOUNT=-1,ERR=9000,READONLY)
      ACCEPT 10020,F2NAM
      F2NAM(30)=0
     OOPEN(UNIT=2,NAME=F2NAM,TYPE='OLD',
     1     BUFFERCOUNT=-1,ERR=9000,READONLY)
      IREC=1
      CALL READF(1,IREC,IBUF,512,IOSB)
      CALL READF(2,IREC,JBUF,512,IOSB)
      N1BLKS=IDBF(9)
      N2BLKS=JDBF(9)
      NSBLKS=N1BLKS
      IF(NSBLKS.GT.N2BLKS)NSBLKS=N2BLKS
      TYPE 10030,NSBLKS
      ACCEPT 10040,NOBLKS
      IF(NOBLKS.LE.0)NOBLKS=NSBLKS
      IF(NOBLKS.GT.NSBLKS)NOBLKS=NSBLKS
      RMS1=0.
      RMS2=0.
      DO 20 IBLK=1,NOBLKS
        IREC=IBLK+1
        CALL READF(1,IREC,IBUF,512,IOSB)
        CALL READF(2,IREC,JBUF,512,IOSB)
        BSUM1=0.
        BSUM2=0.
        DO 10 IDX=1,256
          X=IBUF(IDX)
          BSUM1=BSUM1+X*X
          X=JBUF(IDX)
          BSUM2=BSUM2+X*X
10      CONTINUE
        RMS1=RMS1+BSUM1/256.
        RMS2=RMS2+BSUM2/256.
20    CONTINUE
      RMS1=SQRT(RMS1/NOBLKS)
      RMS2=SQRT(RMS2/NOBLKS)
      DB2=20.*ALOG(RMS2/RMS1)/ALOG(10.)
      TYPE 10050,RMS1,RMS2,DB2
      TYPE 10060
      ACCEPT 10070,GNDB
      TYPE *0075
      ACCEPT 10070,DBMS
      TYPE 10080
      ACCEPT 10020,F3NAM
      F3NAM(30)=0
      IF(.NOT.(F3NAM(1).NE.' '))GO TO 9000 ----------> ABORT
```

```
      COPEN(UNIT=8,NAME=F3NAM,TYPE='NEW',
     1     BUFFERCOUNT=-1,ERR=9000)
      KBUF(1)=1
      KBUF(2)=6400
      SEC=NOBLKS/25.
      MIN=NOBLKS/1500
      ISEC=SEC-60.*MIN
      KBUF(8)=MIN
      KBUF(9)=ISEC
      KBUF(9)=NOBLKS
      IREC=1
      CALL WRITEF(8,IREC,KBUF,512,IOSB)
      RMSN=RMS1*10.**(DBNS/20.)
      GAIN=10.**(GNDB/20.)
      FCTR1=1./(2.+GAIN)
      FCTR2=GAIN*FCTR1
      ISEED1=123
      ISEED2=456
      DO 40 IBLK=1,NOBLKS
        IREC=IBLK+1
        CALL READF(1,IREC,IBUF,512,IOSB)
        CALL READF(2,IREC,JBUF,512,IOSB)
        DO 30 IDX=1,256
          GSNS=0.
          DO 25 KGS=1,12
            GSNS=GSNS+RAN(ISEED1,ISEED2)
   25     CONTINUE
          GSNS=RMSN*(GSNS-6.)
          IF(GSNS.GT.32000.)GSNS=32000.
          IF(GSNS.LT.-32000.)GSNS=-32000.
          KBUF(IDX)=FCTR1*(IBUF(IDX)+GSNS)+FCTR2*JBUF(IDX)
   30   CONTINUE
        CALL WRITEF(8,IREC,KBUF,512,IOSB)
   40 CONTINUE
      TYPE 10090,NOBLKS
 9000 CONTINUE<-------- ABORT
      <-- STOP
10010 FORMAT(' TWO FILES FOR WEIGHTED MERGE WITH NOISE?')
10020 FORMAT(30A1)
10030 FORMAT(' HAVE',I8,' BLOCKS.  HOW MANY TO MERGE?')
10040 FORMAT(I8)
10050 FORMAT(' RMS',2F10.2,', 2ND/1ST:',F10.2,' DB.')
10060 FORMAT(' WHAT DB GAIN FOR 2ND FILE?')
10070 FORMAT(F10.2)
10075 FORMAT(' WHAT DB GAUSS NOISE VS 1ST FILE?(USE .)')
10080 FORMAT(' WHAT OUTPUT FILE?')
10090 FORMAT(' HEADER AND',I8,' DATA BLOCKS WRITTEN.')
      END
```